PHP Handbuch

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Andrei Zmievski

Jouni Ahto

Herausgegeben von

Egon Schmid

Bei der Übersetzung wirkten mit:
Sebastian Bergmann
Friedhelm Betz
Cornelia Boenigk

Wolfgang Drews
Ralf Ebert
Thomas Fromm
Johann-Peter Hartmann
Hartmut Holzgraefe
Martin Jansen
Michael Kaiser
Mark Kronsbein
Hakan Kücükyilmaz
Sven Lauer
Jan Lehnardt
Sebastian Nohn
Catharina Paulsen
Peter Petermann
Georg Richter
Stefan Saasen
Martin Samesch
Egon Schmid
Thomas Schöfbeck
Holger Schranz
Thomas Schürmann
Uwe Steinmann
Christian Ullrich
10-03-2002
Copyright © 1997, 1998, 1999, 2000, 2001, 2002 von der PHP-Dokumentationsgruppe

Copyright

Dieses Handbuch ist urheberrechtlich geschützt. Die Mitglieder der PHP-Dokumentationsgruppe sind auf dem Titelblatt aufgeführt.

Dieses Handbuch kann unter den Bedingungen der GNU General Public License, welche von der Free Software Foundation veröffentlicht wurde, weiterverteilt werden. Es ist entweder die Version 2 oder eine spätere Version der Lizenz zu benutzen.

This manual is © Copyright 1997, 1998, 1999, 2000, 2001, 2002 the PHP Documentation Group. The members of this group are listed on the front page of this manual.

This manual can be redistributed under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

PHP Handbuch

von Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski und Jouni Ahto

von

Herausgegeben von Egon Schmid

von

Bei der Übersetzung wirkten mit:
Sebastian Bergmann
Friedhelm Betz
Cornelia Boenigk
Wolfgang Drews
Ralf Ebert
Thomas Fromm
Johann-Peter Hartmann
Hartmut Holzgraefe
Martin Jansen
Michael Kaiser
Mark Kronsbein
Hakan Kücükyilmaz
Sven Lauer
Jan Lehnardt
Sebastian Nohn
Catharina Paulsen
Peter Petermann
Georg Richter
Stefan Saasen
Martin Samesch
Egon Schmid
Thomas Schöfbeck
Holger Schranz
Thomas Schürmann
Uwe Steinmann
Christian Ullrich

Veröffentlicht 10-03-2002
Copyright © 1997, 1998, 1999, 2000, 2001, 2002 von der PHP-Dokumentationsgruppe

Copyright

Dieses Handbuch ist urheberrechtlich geschützt. Die Mitglieder der PHP-Dokumentationsgruppe sind auf dem Titelblatt aufgeführt.

Dieses Handbuch kann unter den Bedingungen der GNU General Public License, welche von der Free Software Foundation veröffentlicht wurde, weiterverteilt werden. Es ist entweder die Version 2 oder eine spätere Version der Lizenz zu benutzen.

This manual is © Copyright 1997, 1998, 1999, 2000, 2001, 2002 the PHP Documentation Group. The members of this group are listed on the front page of this manual.

This manual can be redistributed under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Inhaltsverzeichnis

Vorwort ..... **i**
Über dieses Handbuch i
I. Einführung **1**
1. Einleitung 1
Was ist PHP? 2
Was kann PHP? 2
2. Installation 5
Download der aktuellsten Version 6
Installation auf UNIX Systemen 6
Apache Modul - Kurzanleitung 6
Building 7
Unix/Linux Installation 7
Pakete verwenden 7
Unix/HP-UX Installationen 7
Unix/Solaris Installation 9
Benötigte Software 9
Pakete verwenden 9
Unix/OpenBSD Installation 10
Ports verwenden 10
Pakete verwenden 10
Unix/Mac OS X Installationen 10
Pakete verwenden 10
Kompilieren für OS X Server 11
Kompilieren für MacOS X Client 12
Komplette Liste der Konfigurationsoptionen 13
Datenbank 13
Ecommerce 18
Graphics 19
Verschiedenes 20
Networking 28
PHP Verhalten 30
Server 30
Text and Sprache 32
XML 32
Installation auf Windows Systemen 33
Windows InstallShield 34
Manuelle Installationsschritte 34
Den Source kompilieren 36
Vorbereitungen 37
Alles zusammenfügen 37
Kompilieren 38
Installation von Windows Erweiterungen 39
Server - Apache 41
Details der Installation von PHP mit Apache auf Unix 41
Installation von PHP auf Windows mit Apache 1.3.x 43
Server - CGI/Commandline 44
Testen 44
Benchmarking 45
Server - fhttpd 45

Server - Caudium ... 45
Servers-IIS/PWS ... 46
Windows and PWS/IIS 3 ... 46
Windows and PWS 4 or newer ... 47
Windows NT/2000 and IIS 4 or newer ... 48
Servers-Netscape and iPlanet ... 48
Installing PHP with Netscape on Sun Solaris ... 49
Installing PHP with Netscape on Windows ... 51
Servers-OmniHTTPd Server ... 52
OmniHTTPd 2.0b1 and up for Windows ... 52
Servers-Oreilly Website Pro ... 53
Oreilly Website Pro 2.5 and up for Windows ... 53
Servers-Xitami ... 53
Xitami for Windows ... 53
Servers-Other web servers ... 53
Problems? ... 54
Read the FAQ ... 54
Other problems ... 54
Bug reports ... 54
3. Konfiguration ... 55
Die Konfigurationsdatei ... 56
Allgemeine Konfigurationseinstellungen ... 56
Mail-Konfigurationsdirektiven ... 60
Safe Mode-Konfigurationsdirektiven ... 61
Debugger-Konfigurationsdirektiven ... 61
Extension Loading-Direktiven ... 61
MySQL-Konfigurationsdirektiven ... 62
mSQL-Konfigurationsdirektiven ... 62
PostgreSQL-Konfigurationsdirektiven ... 62
SESAM Konfigurationsdirektiven ... 63
Sybase-Konfigurationsdirektiven ... 63
Sybase-CT-Konfigurationsdirektiven ... 64
Informix-Konfigurationsdirektiven ... 65
BC Math-Konfigurationsdirektiven ... 66
Browser Capability-Konfigurationsdirektiven ... 66
Unified ODBC-Konfigurationsdirektiven ... 66
4. Sicherheit ... 67
Allgemeine Überlegungen ... 68
CGI-Version ... 69
Mögliche Angriffe ... 69
Fall 1: Nur öffentliche Dateien vorhanden ... 69
Fall 2: --enable-force-cgi-redirect benutzen ... 70
Fall 3: doc_root oder user_dir festlegen ... 70
Fall 4: PHP-Parser außerhalb des Webverzeichnisbaums ... 71
Apache-Modul ... 71
Dateisystem - Sicherheit ... 72
Datenbank - Sicherheit ... 74
Datenbanken designen ... 74
Zur Datenbank verbinden ... 75
Verschlüsseltes Speichermodell ... 75
SQL Injection ... 76
Techniken zur Vermeidung ... 79

Fehlerbehandlung ... 80
Verwendung von Register Globals ... 82
Vom Nutzer übermittelte Daten ... 83
Verstecken von PHP ... 84
Aktuell bleiben ... 85
II. Sprachreferenz ... 86
5. Grundlagen der Syntax ... 86
Den HTML-Bereich der Datei verlassen ... 87
Abgrenzung von Anweisungen ... 87
Kommentare ... 88
6. Typen ... 89
Einführung ... 90
Boolscher Typ ... 90
Syntax ... 91
Umwandlung nach boolean ... 91
Integer Typen ... 92
Syntax ... 92
Integer Überlauf ... 92
Umwandlung nach Integer ... 93
Von Booleans ... 93
Von Fließkomma-Zahlen ... 93
Von Zeichenketten/ Strings ... 94
Von anderen Typen ... 94
Fließkomma-Zahlenwerte ... 94
Strings / Zeichenketten ... 95
Syntax ... 95
Einfache Anführungszeichen (Single quoted) ... 95
Doppelte Anführungszeichen (Double quoted) ... 96
Heredoc ... 97
Variablen-Analyse (parsing) ... 98
Einfache Syntax ... 98
Komplexe (geschweifte) Syntax ... 99
Zugriff auf Zeichen in Strings ... 99
Nützliche Funktionen ... 100
Umwandlung von Zeichenketten / Strings ... 101
Arrays ... 102
Syntax ... 102
Angabe mit array() ... 102
Erzeugen / Verändern mit der eckigen Klammer-Syntax ... 102
Nützliche Funktionen ... 103
Array do’s und don’ts ... 103
Warum ist `$foo[bar]` falsch? ... 103
Aber warum ist es schlecht? ... 104
Beispiele ... 105
Objekte ... 108
Objekt-Initialisierung ... 108
Resource ... 108
Freigabe von Resourcen ... 109
NULL ... 109
Syntax ... 109
Typen-Tricks ... 109

Typ-Umwandlung ... 110
7. Variablen ... 113
Grundlegendes ... 114
Vordefinierte Variablen ... 115
Apache-Variablen ... 115
Umgebungs- / Environment-Variablen ... 117
PHP-Variablen ... 118
Geltungsbereich von Variablen ... 120
Variable Variablen ... 122
Variablen ausserhalb von PHP ... 123
HTML-Formulare (GET and POST) ... 123
IMAGE SUBMIT Variablen-Bezeichner ... 124
HTTP-Cookies ... 125
Umgebungs- / Environment-Variablen ... 125
Punkte in eingelesenen Variablen-Bezeichnern ... 126
Bestimmung des Variablen-Typs ... 126
8. Konstanten ... 127
9. Ausdrücke ... 130
10. Operatoren ... 134
Arithmetische Operatoren ... 135
Zuweisungsoperatoren ... 135
Bit-Operatoren ... 135
Vergleichs-Operatoren ... 136
Fehler-Kontroll-Operatoren ... 137
Operatoren zur Programmausführung ... 137
Inkrementierungs- bzw. Dekrementierungsoperatoren ... 138
Logische Operatoren ... 138
Operator-Rangfolge ... 139
Zeichenketten-Operatoren ... 140
11. Kontroll-Strukturen ... 141
`if` ... 142
`else` ... 142
`elseif` ... 143
Alternative Syntax für Kontroll-Strukturen ... 143
`while` ... 144
`do..while` ... 145
`for` ... 146
`foreach` ... 147
`break` ... 149
`continue` ... 149
`switch` ... 150
require() ... 152
include() ... 154
require_once() ... 156
include_once() ... 159
12. Funktionen ... 160
Vom Nutzer definierte Funktionen ... 161
Funktionsparameter ... 161
Verweise als Parameter übergeben ... 161
Vorgabewerte für Parameter ... 162
Rückgabewerte ... 163
`old_function` ... 163

Variablenfunktionen ... 164
13. Klassen und Objekte ... 165
`Klassen` ... 166
14. Referenzen in PHP ... 169
Was Referenzen sind ... 170
Was Referenzen leisten ... 170
Was Referenzen nicht sind ... 171
Referenzen zurückgeben ... 171
Referenzen aufheben ... 172
Referenzen entdecken ... 172
`global` References ... 172
`$this` ... 172
III. Features ... 174
15. Fehlerbehandlung ... 174
16. Erzeugen und Verändern von Bildern ... 176
17. HTTP-Authentifizierung mit PHP ... 178
18. Cookies ... 181
19. Steuerung von Dateiuploads ... 183
Dateiuploads mit POST ... 184
Häufige Probleme ... 184
Mehrere Dateien uploaden ... 185
PUT-Unterstützung ... 185
20. Zugriff auf entfernte Dateien ... 187
21. Verbindungssteuerung ... 190
22. Persistente Datenbankverbindungen ... 192
23. Safe Mode ... 195
Funktionen, die durch Safe Mode deaktiviert oder eingeschränkt werden ... 197
IV. Funktionsreferenz ... 201
I. Apache-spezifische Funktionen ... 201
apache_lookup_uri ... 202
apache_note ... 202
ascii2ebcdic ... 202
ebcdic2ascii ... 203
getallheaders ... 203
virtual ... 204
apache_child_terminate ... 204
apache_setenv ... 204
II. Array Funktionen ... 205
array ... 206
array_change_key_case ... 207
array_chunk ... 208
array_count_values ... 209
array_diff ... 210
array_filter ... 210
array_flip ... 211
array_fill ... 212
array_intersect ... 213
array_key_exists ... 214
array_keys ... 214
array_map ... 216
array_merge ... 219

array_merge_recursive 220
array_multisort 221
array_pad 222
array_pop 222
array_push 223
array_rand 224
array_reverse 225
array_reduce 226
array_shift 226
array_slice 227
array_splice 228
array_sum 229
array_unique 229
array_unshift 231
array_values 231
array_walk 232
arsort 234
asort 235
compact 235
count 236
current 237
each 238
end 239
extract 239
in_array 241
array_search 243
key 243
krsort 243
ksort 244
list 245
natsort 246
natcasesort 247
next 247
pos 247
prev 248
range 248
reset 249
rsort 249
shuffle 250
sizeof 250
sort 251
uasort 252
uksort 252
usort 253
III. Aspell Funktionen [veraltet] 256
aspell_new 257
aspell_check 257
aspell_check-raw 257
aspell_suggest 258
IV. Mathematische Funktionen mit beliebiger Genauigkeit 259
bcadd 260
bccomp 260

bcdiv ... 260
bcmod ... 260
bcmul ... 260
bcpow ... 261
bcscale ... 261
bcsqrt ... 261
bcsub ... 261
V. Bzip2 Komprimierungsfunktionen ... 263
bzclose ... 264
bzcompress ... 264
bzdecompress ... 264
bzerrno ... 265
bzerror ... 265
bzerrstr ... 266
bzflush ... 266
bzopen ... 266
bzread ... 267
bzwrite ... 267
VI. Kalender-Funktionen ... 269
JDToGregorian ... 270
GregorianToJD ... 270
JDToJulian ... 270
JulianToJD ... 270
JDToJewish ... 271
JewishToJD ... 271
JDToFrench ... 271
FrenchToJD ... 271
JDMonthName ... 272
JDDayOfWeek ... 272
easter_date ... 273
easter_days ... 273
unixtojd ... 274
jdtounix ... 274
VII. CCVS API Funktionen ... 275
ccvs_init ... 276
ccvs_done ... 276
ccvs_new ... 276
ccvs_add ... 276
ccvs_delete ... 277
ccvs_auth ... 277
ccvs_return ... 277
ccvs_reverse ... 278
ccvs_sale ... 278
ccvs_void ... 278
ccvs_status ... 278
ccvs_count ... 279
ccvs_lookup ... 279
ccvs_report ... 279
ccvs_command ... 280
ccvs_textvalue ... 280
VIII. COM Support Funktionen für Windows ... 281
COM ... 282

VARIANT ... 283
com_load ... 284
com_invoke ... 284
com_propget ... 284
com_get ... 285
com_propput ... 285
com_propset ... 285
com_set ... 285
com_addref ... 285
com_release ... 286
com_isenum ... 286
com_load_typelib ... 286
IX. Klassen- und Objekt-Funktionen ... 287
call_user_method ... 290
class_exists ... 290
get_class ... 290
get_class_methods ... 291
get_class_vars ... 291
get_declared_classes ... 291
get_object_vars ... 291
get_parent_class ... 293
is_subclass_of ... 293
method_exists ... 293
X. ClibPDF Funktionen ... 294
cpdf_add_annotation ... 297
cpdf_add_outline ... 297
cpdf_arc ... 297
cpdf_begin_text ... 298
cpdf_circle ... 298
cpdf_clip ... 298
cpdf_close ... 299
cpdf_closepath ... 299
cpdf_closepath_fill_stroke ... 299
cpdf_closepath_stroke ... 299
cpdf_continue_text ... 299
cpdf_curveto ... 300
cpdf_end_text ... 300
cpdf_fill ... 300
cpdf_fill_stroke ... 301
cpdf_finalize ... 301
cpdf_finalize_page ... 301
cpdf_global_set_document_limits ... 301
cpdf_import_jpeg ... 302
cpdf_lineto ... 302
cpdf_moveto ... 302
cpdf_newpath ... 303
cpdf_open ... 303
cpdf_output_buffer ... 303
cpdf_page_init ... 304
cpdf_place_inline_image ... 304
cpdf_rect ... 304
cpdf_restore ... 305

cpdf_rlineto ... 305
cpdf_rmoveto ... 305
cpdf_rotate ... 306
cpdf_rotate_text ... 306
cpdf_save ... 306
cpdf_save_to_file ... 306
cpdf_scale ... 307
cpdf_set_action_url ... 307
cpdf_set_char_spacing ... 307
cpdf_set_creator ... 307
cpdf_set_current_page ... 308
cpdf_set_font ... 308
cpdf_set_font_directories ... 308
cpdf_set_font_map_file ... 308
cpdf_set_horiz_scaling ... 309
cpdf_set_keywords ... 309
cpdf_set_leading ... 309
cpdf_set_page_animation ... 309
cpdf_set_subject ... 310
cpdf_set_text_matrix ... 310
cpdf_set_text_pos ... 310
cpdf_set_text_rendering ... 310
cpdf_set_text_rise ... 311
cpdf_set_title ... 311
cpdf_set_viewer_preferences ... 311
cpdf_set_word_spacing ... 311
cpdf_setdash ... 312
cpdf_setflat ... 312
cpdf_setgray ... 312
cpdf_setgray_fill ... 312
cpdf_setgray_stroke ... 313
cpdf_setlinecap ... 313
cpdf_setlinejoin ... 313
cpdf_setlinewidth ... 313
cpdf_setmiterlimit ... 313
cpdf_setrgbcolor ... 313
cpdf_setrgbcolor_fill ... 314
cpdf_setrgbcolor_stroke ... 314
cpdf_show ... 314
cpdf_show_xy ... 314
cpdf_stringwidth ... 315
cpdf_stroke ... 315
cpdf_text ... 315
cpdf_translate ... 316
XI. Crack Funktionen ... 317
crack_opendict ... 318
crack_closedict ... 318
crack_check ... 318
crack_getlastmessage ... 319
XII. CURL, Client URL Library Functions ... 320
curl_init ... 321
curl_setopt ... 321

curl_exec ... 324
curl_close ... 324
curl_version ... 325
curl_errno ... 325
curl_error ... 325
curl_getinfo ... 325
XIII. Cybercash Bezahl Funktionen ... 327
cybercash_encr ... 328
cybercash_decr ... 328
cybercash_base64_encode ... 328
cybercash_base64_decode ... 328
XIV. Crédit Mutuel CyberMUT functions ... 329
cybermut_creerformulairecm ... 330
cybermut_testmac ... 330
cybermut_creerreponsecm ... 331
XV. Cyrus IMAP administration functions ... 332
cyrus_connect ... 333
cyrus_authenticate ... 333
cyrus_bind ... 333
cyrus_unbind ... 333
cyrus_query ... 334
cyrus_close ... 334
XVI. Character type functions ... 335
ctype_alnum ... 336
ctype_alpha ... 336
ctype_cntrl ... 336
ctype_digit ... 336
ctype_lower ... 336
ctype_graph ... 337
ctype_print ... 337
ctype_punct ... 337
ctype_space ... 337
ctype_upper ... 338
ctype_xdigit ... 338
XVII. dba Datenbank (dbm-style) Abstraktions-Funktionen ... 339
dba_close ... 342
dba_delete ... 342
dba_exists ... 342
dba_fetch ... 342
dba_firstkey ... 343
dba_insert ... 343
dba_nextkey ... 343
dba_popen ... 344
dba_open ... 345
dba_optimize ... 345
dba_replace ... 345
dba_sync ... 346
XVIII. Datums- und Zeit-Funktionen ... 347
checkdate ... 348
date ... 348
getdate ... 349
gettimeofday ... 350

gmdate ... 350
gmmktime ... 351
gmstrftime ... 351
localtime ... 351
microtime ... 352
mktime ... 352
strftime ... 353
time ... 355
strtotime ... 355
XIX. dBase Funktionen ... 356
dbase_create ... 357
dbase_open ... 358
dbase_close ... 358
dbase_pack ... 358
dbase_add_record ... 358
dbase_replace_record ... 358
dbase_delete_record ... 359
dbase_get_record ... 359
dbase_get_record_with_names ... 359
dbase_numfields ... 359
dbase_numrecords ... 360
XX. DBM Datenbankfunktionen ... 361
dbmopen ... 362
dbmclose ... 362
dbmexists ... 362
dbmfetch ... 363
dbminsert ... 363
dbmreplace ... 363
dbmdelete ... 364
dbmfirstkey ... 364
dbmnextkey ... 364
dblist ... 364
XXI. dbx Funktionen ... 365
dbx_close ... 366
dbx_connect ... 366
dbx_error ... 367
dbx_query ... 368
dbx_sort ... 371
dbx_compare ... 372
XXII. DB++ Functions ... 374
dbplus_add ... 377
dbplus_aql ... 377
dbplus_chdir ... 377
dbplus_close ... 378
dbplus_curr ... 378
dbplus_errcode ... 378
dbplus_errno ... 379
dbplus_find ... 379
dbplus_first ... 380
dbplus_flush ... 380
dbplus_freealllocks ... 380
dbplus_freelock ... 381

dbplus_freerlocks ... 381
dbplus_getlock ... 382
dbplus_getunique ... 382
dbplus_info ... 382
dbplus_last ... 383
dbplus_lockrel ... 383
dbplus_next ... 383
dbplus_open ... 384
dbplus_prev ... 384
dbplus_rchperm ... 385
dbplus_rcreate ... 385
dbplus_rcrtexact ... 386
dbplus_rcrtlike ... 386
dbplus_resolve ... 386
dbplus_rkeys ... 387
dbplus_restorepos ... 387
dbplus_ropen ... 387
dbplus_rquery ... 388
dbplus_rrename ... 388
dbplus_rsecindex ... 388
dbplus_runlink ... 389
dbplus_rzap ... 389
dbplus_savepos ... 389
dbplus_setindex ... 390
dbplus_setindexbynumber ... 390
dbplus_sql ... 390
dbplus_tcl ... 391
dbplus_tremove ... 391
dbplus_undo ... 392
dbplus_undoprepare ... 392
dbplus_unlockrel ... 392
dbplus_unselect ... 393
dbplus_update ... 393
dbplus_xlockrel ... 393
dbplus_xunlockrel ... 394
XXIII. Direkte Ein-/Ausgabe Funktionen ... 395
dio_open ... 396
dio_read ... 396
dio_write ... 396
dio_truncate ... 396
dio_stat ... 397
dio_seek ... 397
dio_fcntl ... 398
dio_close ... 398
XXIV. Verzeichnis-Funktionen ... 399
chdir ... 400
dir ... 400
closedir ... 400
getcwd ... 400
opendir ... 401
readdir ... 401
rewinddir ... 402

XXV. DOM XML Funktionen ... 403
xmldoc ... 404
xmldocfile ... 404
xmltree ... 404
XXVI. .NET Funktionen ... 405
dotnet_load ... 406
XXVII. Error Handling and Logging Functions ... 407
error_log ... 408
error_reporting ... 409
restore_error_handler ... 409
set_error_handler ... 409
trigger_error ... 412
user_error ... 412
XXVIII. FrontBase Functions ... 413
fbsql_affected_rows ... 414
fbsql_autocommit ... 414
fbsql_change_user ... 414
fbsql_close ... 415
fbsql_commit ... 415
fbsql_connect ... 415
fbsql_create_db ... 416
fbsql_create_blob ... 416
fbsql_create_clob ... 417
fbsql_database_password ... 418
fbsql_data_seek ... 418
fbsql_db_query ... 419
fbsql_db_status ... 419
fbsql_drop_db ... 420
fbsql_errno ... 420
fbsql_error ... 421
fbsql_fetch_array ... 421
fbsql_fetch_assoc ... 422
fbsql_fetch_field ... 423
fbsql_fetch_lengths ... 424
fbsql_fetch_object ... 424
fbsql_fetch_row ... 425
fbsql_field_flags ... 425
fbsql_field_name ... 425
fbsql_field_len ... 426
fbsql_field_seek ... 426
fbsql_field_table ... 427
fbsql_field_type ... 427
fbsql_free_result ... 428
fbsql_insert_id ... 428
fbsql_list_dbs ... 428
fbsql_list_fields ... 429
fbsql_list_tables ... 430
fbsql_next_result ... 430
fbsql_num_fields ... 430
fbsql_num_rows ... 431
fbsql_pconnect ... 431
fbsql_query ... 432

fbsql_read_blob ... 433
fbsql_read_clob ... 434
fbsql_result ... 434
fbsql_rollback ... 435
fbsql_set_lob_mode ... 435
fbsql_select_db ... 435
fbsql_start_db ... 436
fbsql_stop_db ... 436
fbsql_tablename ... 436
fbsql_warnings ... 437
fbsql_database ... 437
fbsql_get_autostart_info ... 437
fbsql_hostname ... 438
fbsql_password ... 438
fbsql_set_transaction ... 438
fbsql_username ... 439
XXIX. filePro Funktionen ... 440
filepro ... 441
filepro_fieldname ... 441
filepro_fieldtype ... 441
filepro_fieldwidth ... 441
filepro_retrieve ... 441
filepro_fieldcount ... 441
filepro_rowcount ... 442
XXX. Funktionen des Dateisystems ... 443
basename ... 444
chgrp ... 444
chmod ... 444
chown ... 445
clearstatcache ... 445
copy ... 446
delete ... 446
dirname ... 446
disk_free_space ... 447
diskfreespace ... 447
disk_total_space ... 447
fclose ... 448
feof ... 448
fflush ... 448
fgetc ... 449
fgetcsv ... 449
fgets ... 450
fgetss ... 450
file ... 451
file_exists ... 452
fileatime ... 452
filectime ... 452
filegroup ... 453
fileinode ... 453
filemtime ... 454
fileowner ... 454
fileperms ... 454

filesize ... 455
filetype ... 455
flock ... 455
fopen ... 456
fpassthru ... 458
fputs ... 458
fread ... 459
fscanf ... 459
fseek ... 460
fstat ... 461
ftell ... 461
ftruncate ... 461
fwrite ... 462
set_file_buffer ... 462
is_dir ... 463
is_executable ... 463
is_file ... 463
is_link ... 464
is_readable ... 464
is_writable ... 464
is_writeable ... 465
is_uploaded_file ... 465
link ... 466
linkinfo ... 466
mkdir ... 466
move_uploaded_file ... 467
parse_ini_file ... 468
pathinfo ... 469
pclose ... 470
popen ... 470
readfile ... 471
readlink ... 471
rename ... 472
rewind ... 472
rmdir ... 472
stat ... 472
lstat ... 473
realpath ... 474
symlink ... 474
tempnam ... 475
tmpfile ... 475
touch ... 476
umask ... 477
unlink ... 477
XXXI. Forms Data Format Funktionen ... 478
fdf_open ... 480
fdf_close ... 480
fdf_create ... 480
fdf_save ... 481
fdf_get_value ... 481
fdf_set_value ... 481
fdf_next_field_name ... 482

fdf_set_ap ... 482
fdf_set_status ... 482
fdf_get_status ... 482
fdf_set_file ... 482
fdf_get_file ... 483
fdf_set_flags ... 483
fdf_set_opt ... 483
fdf_set_submit_form_action ... 483
fdf_set_javascript_action ... 484
fdf_set_encoding ... 484
fdf_add_template ... 484
XXXII. FriBiDi functions ... 485
fribidi_log2vis ... 486
XXXIII. FTP-Funktionen ... 487
ftp_connect ... 488
ftp_login ... 488
ftp_pwd ... 488
ftp_cdup ... 488
ftp_chdir ... 488
ftp_mkdir ... 489
ftp_rmdir ... 489
ftp_nlist ... 489
ftp_rawlist ... 489
ftp_systype ... 489
ftp_pasv ... 490
ftp_get ... 490
ftp_fget ... 490
ftp_put ... 490
ftp_fput ... 491
ftp_size ... 491
ftp_mdtm ... 491
ftp_rename ... 491
ftp_delete ... 492
ftp_site ... 492
ftp_quit ... 492
XXXIV. Function Handling functions ... 493
call_user_func ... 494
create_function ... 494
func_get_arg ... 496
func_get_args ... 497
func_num_args ... 497
function_exists ... 498
register_shutdown_function ... 498
XXXV. GNU Gettext ... 499
bindtextdomain ... 500
dcgettext ... 500
dgettext ... 500
gettext ... 500
textdomain ... 501
XXXVI. GMP functions ... 502
gmp_init ... 503
gmp_intval ... 503

gmp_strval ... 503
gmp_add ... 504
gmp_sub ... 504
gmp_mul ... 504
gmp_div_q ... 504
gmp_div_r ... 505
gmp_div_qr ... 505
gmp_div ... 505
gmp_mod ... 506
gmp_divexact ... 506
gmp_cmp ... 506
gmp_neg ... 506
gmp_com ... 506
gmp_abs ... 507
gmp_sign ... 507
gmp_fact ... 507
gmp_sqrt ... 507
gmp_sqrtrm ... 507
gmp_perfect_square ... 508
gmp_pow ... 508
gmp_powm ... 508
gmp_prob_prime ... 508
gmp_gcd ... 508
gmp_gcdext ... 509
gmp_invert ... 509
gmp_legendre ... 509
gmp_jacobi ... 509
gmp_random ... 509
gmp_and ... 510
gmp_or ... 510
gmp_xor ... 510
gmp_setbit ... 510
gmp_clrbit ... 510
gmp_scan0 ... 511
gmp_scan1 ... 511
gmp_popcount ... 511
gmp_hamdist ... 511
XXXVII. HTTP-Funktionen ... 512
header ... 513
headers_sent ... 513
setcookie ... 514
XXXVIII. Hyperwave functions ... 516
hw_Array2Objrec ... 520
hw_Children ... 520
hw_ChildrenObj ... 520
hw_Close ... 520
hw_Connect ... 520
hw_Cp ... 521
hw_Deleteobject ... 521
hw_DocByAnchor ... 521
hw_DocByAnchorObj ... 521
hw_Document_Attributes ... 521

hw_Document_BodyTag 522
hw_Document_Content 522
hw_Document_SetContent 522
hw_Document_Size 523
hw_ErrorMsg 523
hw_EditText 523
hw_Error 523
hw_Free_Document 523
hw_GetParents 524
hw_GetParentsObj 524
hw_GetChildColl 524
hw_GetChildCollObj 524
hw_GetRemote 524
hw_GetRemoteChildren 525
hw_GetSrcByDestObj 525
hw_GetObject 525
hw_GetAndLock 526
hw_GetText 526
hw_GetObjectByQuery 527
hw_GetObjectByQueryObj 527
hw_GetObjectByQueryColl 527
hw_GetObjectByQueryCollObj 528
hw_GetChildDocColl 528
hw_GetChildDocCollObj 528
hw_GetAnchors 528
hw_GetAnchorsObj 529
hw_Mv 529
hw_Identify 529
hw_InCollections 529
hw_Info 530
hw_InsColl 530
hw_InsDoc 530
hw_InsertDocument 530
hw_InsertObject 531
hw_mapid 531
hw_Modifyobject 531
hw_New_Document 534
hw_Objrec2Array 534
hw_Output_Document 534
hw_pConnect 534
hw_PipeDocument 535
hw_Root 535
hw_Unlock 535
hw_Who 535
hw_getusername 536
hw_stat 536
hw_setlinkroot 536
hw_connection_info 536
hw_dummy 537
hw_insertanchors 537
hw_getrellink 537
hw_changeobject 538

XXXIX. ICAP Functions [deprecated] ... 539
icap_open ... 540
icap_close ... 540
icap_fetch_event ... 540
icap_list_events ... 541
icap_store_event ... 541
icap_delete_event ... 542
icap_snooze ... 542
icap_list_alarms ... 542
icap_create_calendar ... 543
icap_rename_calendar ... 543
icap_delete_calendar ... 543
icap_reopen ... 543
XL. iconv functions ... 545
iconv ... 546
iconv_get_encoding ... 546
iconv_set_encoding ... 546
ob_iconv_handler ... 546
XLI. Grafik-Funktionen ... 548
GetImageSize ... 549
ImageArc ... 549
ImageChar ... 550
ImageCharUp ... 550
ImageColorAllocate ... 550
ImageColorAt ... 550
ImageColorClosest ... 551
ImageColorDeAllocate ... 551
ImageColorExact ... 551
ImageColorResolve ... 551
ImageColorSet ... 552
ImageColorsForIndex ... 552
ImageColorsTotal ... 552
ImageColorTransparent ... 552
ImageCopy ... 553
ImageCopyResized ... 553
ImageCreate ... 553
ImageCreateFromGIF ... 554
ImageCreateFromJPEG ... 554
ImageCreateFromPNG ... 555
ImageDashedLine ... 556
ImageDestroy ... 556
ImageFill ... 556
ImageFilledPolygon ... 556
ImageFilledRectangle ... 556
ImageFillToBorder ... 557
ImageFontHeight ... 557
ImageFontWidth ... 557
ImageGammaCorrect ... 557
ImageGIF ... 557
ImageJPEG ... 559
ImagePNG ... 559
ImageInterlace ... 560

ImageLine ... 560
ImageLoadFont ... 560
ImagePolygon ... 561
ImagePSBBox ... 561
ImagePSEncodeFont ... 562
ImagePsExtendFont ... 562
ImagePSFreeFont ... 562
ImagePSLoadFont ... 562
ImagePsSlantFont ... 563
ImagePSText ... 563
ImageRectangle ... 564
ImageSetPixel ... 564
ImageString ... 564
ImageStringUp ... 564
ImageSX ... 565
ImageSY ... 565
ImageTTFBBox ... 565
ImageTTFText ... 566
ImageTypes ... 567
read_exif_data ... 567
XLII. IMAP, POP3 und NNTP Funktionen ... 569
imap_append ... 570
imap_base64 ... 570
imap_body ... 570
imap_check ... 571
imap_close ... 572
imap_createmailbox ... 572
imap_delete ... 573
imap_deletemailbox ... 574
imap_expunge ... 574
imap_fetchbody ... 574
imap_fetchstructure ... 575
imap_header ... 576
imap_headers ... 578
imap_listmailbox ... 578
imap_getmailboxes ... 578
imap_listsubscribed ... 579
imap_getsubscribed ... 580
imap_mail_copy ... 580
imap_mail_move ... 580
imap_num_msg ... 581
imap_num_recent ... 581
imap_open ... 581
imap_ping ... 582
imap_renamemailbox ... 583
imap_reopen ... 583
imap_subscribe ... 583
imap_undelete ... 584
imap_unsubscribe ... 584
imap_qprint ... 584
imap_8bit ... 584
imap_binary ... 584

imap_scanmailbox ... 585
imap_mailboxmsginfo ... 585
imap_rfc822_write_address ... 586
imap_rfc822_parse_adrlist ... 586
imap_setflag_full ... 587
imap_clearflag_full ... 587
imap_sort ... 588
imap_fetchheader ... 588
imap_uid ... 589
imap_msgno ... 589
imap_search ... 589
imap_last_error ... 590
imap_errors ... 590
imap_alerts ... 591
imap_status ... 591
imap_utf7_decode ... 592
imap_utf7_encode ... 592
imap_utf8 ... 592
imap_fetch_overview ... 593
imap_mime_header_decode ... 594
imap_mail_compose ... 594
imap_mail ... 594
XLIII. Informix functions ... 596
ifx_connect ... 598
ifx_pconnect ... 598
ifx_close ... 598
ifx_query ... 599
ifx_prepare ... 600
ifx_do ... 601
ifx_error ... 601
ifx_errormsg ... 602
ifx_affected_rows ... 602
ifx_getsqlca ... 603
ifx_fetch_row ... 603
ifx_htmltbl_result ... 604
ifx_fieldtypes ... 605
ifx_fieldproperties ... 606
ifx_num_fields ... 606
ifx_num_rows ... 606
ifx_free_result ... 607
ifx_create_char ... 607
ifx_free_char ... 607
ifx_update_char ... 607
ifx_get_char ... 607
ifx_create_blob ... 607
ifx_copy_blob ... 608
ifx_free_blob ... 608
ifx_get_blob ... 608
ifx_update_blob ... 608
ifx_blobinfile_mode ... 609
ifx_textasvarchar ... 609
ifx_byteasvarchar ... 609

ifx_nullformat ... 609
ifxus_create_slob ... 609
ifxus_free_slob ... 610
ifxus_close_slob ... 610
ifxus_open_slob ... 610
ifxus_tell_slob ... 610
ifxus_seek_slob ... 610
ifxus_read_slob ... 611
ifxus_write_slob ... 611
XLIV. InterBase-Funktionen ... 612
ibase_connect ... 613
ibase_pconnect ... 613
ibase_close ... 614
ibase_query ... 614
ibase_fetch_row ... 614
ibase_fetch_object ... 615
ibase_free_result ... 615
ibase_prepare ... 615
ibase_execute ... 615
ibase_free_query ... 616
ibase_timefmt ... 616
ibase_num_fields ... 617
XLV. Ingres II Funktionen ... 618
ingres_connect ... 619
ingres_pconnect ... 619
ingres_close ... 620
ingres_query ... 620
ingres_num_rows ... 622
ingres_num_fields ... 622
ingres_field_name ... 623
ingres_field_type ... 623
ingres_field_nullable ... 624
ingres_field_length ... 624
ingres_field_precision ... 624
ingres_field_scale ... 625
ingres_fetch_array ... 625
ingres_fetch_row ... 626
ingres_fetch_object ... 627
ingres_rollback ... 628
ingres_commit ... 628
ingres_autocommit ... 629
XLVI. IRC Gateway Funktionen ... 630
ircg_pconnect ... 631
ircg_fetch_error_msg ... 631
ircg_set_current ... 631
ircg_join ... 632
ircg_part ... 632
ircg_msg ... 632
ircg_notice ... 632
ircg_nick ... 633
ircg_topic ... 633
ircg_channel_mode ... 633

ircg_html_encode ... 633
ircg_whois ... 634
ircg_kick ... 634
ircg_ignore_add ... 634
ircg_ignore_del ... 634
ircg_disconnect ... 634
ircg_is_conn_alive ... 635
ircg_lookup_format_messages ... 635
ircg_register_format_messages ... 635
ircg_set_on_die ... 636
ircg_set_file ... 637
ircg_get_username ... 637
ircg_nickname_escape ... 637
ircg_nickname_unescape ... 637
XLVII. Java ... 639
java_last_exception_clear ... 641
java_last_exception_get ... 641
XLVIII. LDAP Funktionen ... 642
ldap_add ... 645
ldap_bind ... 645
ldap_close ... 646
ldap_compare ... 646
ldap_connect ... 647
ldap_count_entries ... 648
ldap_delete ... 648
ldap_dn2ufn ... 648
ldap_err2str ... 648
ldap_errno ... 649
ldap_error ... 650
ldap_explode_dn ... 650
ldap_first_attribute ... 650
ldap_first_entry ... 651
ldap_free_result ... 651
ldap_get_attributes ... 651
ldap_get_dn ... 652
ldap_get_entries ... 653
ldap_get_option ... 653
ldap_get_values ... 654
ldap_get_values_len ... 655
ldap_list ... 655
ldap_modify ... 656
ldap_mod_add ... 656
ldap_mod_del ... 657
ldap_mod_replace ... 657
ldap_next_attribute ... 657
ldap_next_entry ... 658
ldap_read ... 658
ldap_rename ... 658
ldap_search ... 659
ldap_set_option ... 661
ldap_unbind ... 662
ldap_8859_to_t61 ... 662

ldap_first_reference ... 662
ldap_next_reference ... 663
ldap_parse_reference ... 663
ldap_parse_result ... 663
ldap_set_rebind_proc ... 664
ldap_sort ... 664
ldap_start_tls ... 664
ldap_t61_to_8859 ... 665
XLIX. Mail Funktionen ... 666
mail ... 667
L. mailparse functions ... 668
mailparse_uudecode_all ... 669
mailparse_rfc822_parse_addresses ... 669
mailparse_determine_best_xfer_encoding ... 669
mailparse_stream_encode ... 670
mailparse_msg_parse ... 670
mailparse_msg_parse_file ... 671
mailparse_msg_free ... 671
mailparse_msg_create ... 672
mailparse_msg_get_structure ... 672
mailparse_msg_extract_part ... 673
mailparse_msg_extract_part_file ... 673
mailparse_msg_get_part_data ... 674
mailparse_msg_get_part ... 674
LI. Mathematische Funktionen ... 676
abs ... 677
acos ... 677
asin ... 677
atan ... 677
atan2 ... 677
base_convert ... 678
BinDec ... 678
ceil ... 678
cos ... 678
DecBin ... 679
DecHex ... 679
DecOct ... 679
deg2rad ... 679
exp ... 680
floor ... 680
getrandmax ... 680
HexDec ... 680
log ... 680
log10 ... 681
max ... 681
min ... 681
mt_rand ... 682
mt_srand ... 682
mt_getrandmax ... 683
number_format ... 683
OctDec ... 683
pi ... 683

pow ... 684
rad2deg ... 684
rand ... 684
round ... 684
sin ... 685
sqrt ... 685
srand ... 685
tan ... 686
LII. Multi-Byte String Functions ... 687
mb_language ... 694
mb_parse_str ... 694
mb_internal_encoding ... 694
mb_http_input ... 695
mb_http_output ... 695
mb_detect_order ... 695
mb_substitute_character ... 696
mb_output_handler ... 697
mb_preferred_mime_name ... 698
mb_strlen ... 698
mb_strpos ... 699
mb_strrpos ... 699
mb_substr ... 699
mb_strcut ... 700
mb_strwidth ... 700
mb_strimwidth ... 701
mb_convert_encoding ... 701
mb_detect_encoding ... 702
mb_convert_kana ... 702
mb_encode_mimeheader ... 703
mb_decode_mimeheader ... 704
mb_convert_variables ... 704
mb_encode_numericentity ... 705
mb_decode_numericentity ... 706
mb_send_mail ... 706
mb_get_info ... 707
mb_regex_encoding ... 707
mb_ereg ... 708
mb_eregi ... 708
mb_ereg_replace ... 709
mb_eregi_replace ... 710
mb_split ... 710
mb_ereg_match ... 711
mb_ereg_search ... 711
mb_ereg_search_pos ... 712
mb_ereg_search_regs ... 712
mb_ereg_search_init ... 713
mb_ereg_search_getregs ... 713
mb_ereg_search_getpos ... 714
mb_ereg_search_setpos ... 714
LIII. MCAL Funktionen ... 716
mcal_open ... 717
mcal_close ... 717

mcal_fetch_event ... 717
mcal_list_events ... 718
mcal_append_event ... 718
mcal_store_event ... 718
mcal_delete_event ... 718
mcal_snooze ... 719
mcal_list_alarms ... 719
mcal_event_init ... 719
mcal_event_set_category ... 719
mcal_event_set_title ... 720
mcal_event_set_description ... 720
mcal_event_set_start ... 720
mcal_event_set_end ... 720
mcal_event_set_alarm ... 720
mcal_event_set_class ... 721
mcal_is_leap_year ... 721
mcal_days_in_month ... 721
mcal_date_valid ... 721
mcal_time_valid ... 721
mcal_day_of_week ... 722
mcal_day_of_year ... 722
mcal_date_compare ... 722
mcal_next_recurrence ... 722
mcal_event_set_recur_none ... 722
mcal_event_set_recur_daily ... 723
mcal_event_set_recur_weekly ... 723
mcal_event_set_recur_monthly_mday ... 723
mcal_event_set_recur_monthly_wday ... 723
mcal_event_set_recur_yearly ... 723
mcal_fetch_current_stream_event ... 724
LIV. Mcrypt Encryption Functions ... 725
mcrypt_get_cipher_name ... 729
mcrypt_get_block_size ... 729
mcrypt_get_key_size ... 729
mcrypt_create_iv ... 730
mcrypt_cbc ... 730
mcrypt_cfb ... 731
mcrypt_ecb ... 731
mcrypt_ofb ... 732
mcrypt_list_algorithms ... 732
mcrypt_list_modes ... 732
mcrypt_get_iv_size ... 733
mcrypt_encrypt ... 733
mcrypt_decrypt ... 734
mcrypt_module_open ... 735
mcrypt_module_close ... 735
mcrypt_generic_deinit ... 736
mcrypt_generic_init ... 736
mcrypt_generic ... 736
mdecrypt_generic ... 737
mcrypt_generic_end ... 737
mcrypt_enc_self_test ... 737

mcrypt_enc_is_block_algorithm_mode ... 738
mcrypt_enc_is_block_algorithm ... 738
mcrypt_enc_is_block_mode ... 738
mcrypt_enc_get_block_size ... 738
mcrypt_enc_get_key_size ... 738
mcrypt_enc_get_supported_key_sizes ... 739
mcrypt_enc_get_iv_size ... 739
mcrypt_enc_get_algorithms_name ... 739
mcrypt_enc_get_modes_name ... 739
mcrypt_module_self_test ... 739
mcrypt_module_is_block_algorithm_mode ... 740
mcrypt_module_is_block_algorithm ... 740
mcrypt_module_is_block_mode ... 740
mcrypt_module_get_algo_block_size ... 740
mcrypt_module_get_algo_key_size ... 741
mcrypt_module_get_supported_key_sizes ... 741
LV. Hash Funktionen ... 742
mhash_get_hash_name ... 744
mhash_get_block_size ... 744
mhash_count ... 744
mhash ... 745
LVI. Microsoft SQL Server Funktionen ... 746
mssql_close ... 747
mssql_connect ... 747
mssql_data_seek ... 747
mssql_fetch_array ... 747
mssql_fetch_field ... 748
mssql_fetch_object ... 748
mssql_fetch_row ... 749
mssql_field_length ... 749
mssql_field_name ... 749
mssql_field_seek ... 749
mssql_field_type ... 750
mssql_free_result ... 750
mssql_get_last_message ... 750
mssql_min_error_severity ... 750
mssql_min_message_severity ... 750
mssql_next_result ... 751
mssql_num_fields ... 751
mssql_num_rows ... 751
mssql_pconnect ... 752
mssql_query ... 752
mssql_result ... 752
mssql_select_db ... 753
mssql_bind ... 753
mssql_execute ... 753
mssql_fetch_assoc ... 754
mssql_fetch_batch ... 754
mssql_guid_string ... 754
mssql_init ... 755
mssql_rows_affected ... 755
LVII. Ming functions for Flash ... 756

ming_setcubicthreshold 758
ming_setscale 758
ming_useswfversion 758
swfbutton_keypress 758
SWFMovie 759
SWFMovie->output 759
SWFMovie->save 760
SWFMovie->add 760
SWFMovie->remove 760
SWFMovie->setbackground 761
SWFMovie->setrate 761
SWFMovie->setdimension 762
SWFMovie->setframes 762
SWFMovie->nextframe 762
SWFMovie->streammp3 763
SWFDisplayItem 763
SWFDisplayItem->moveTo 764
SWFDisplayItem->move 764
SWFDisplayItem->scaleTo 765
SWFDisplayItem->scale 765
SWFDisplayItem->rotateTo 765
SWFDisplayItem->Rotate 767
SWFDisplayItem->skewXTo 768
SWFDisplayItem->skewX 768
SWFDisplayItem->skewYTo 769
SWFDisplayItem->skewY 769
SWFDisplayItem->setDepth 769
SWFDisplayItem->remove 770
SWFDisplayItem->setName 770
SWFDisplayItem->setRatio 771
SWFDisplayItem->addColor 772
SWFDisplayItem->multColor 773
SWFShape 773
SWFShape->setLine 774
SWFShape->addFill 776
SWFShape->setLeftFill 778
SWFShape->setRightFill 778
SWFShape->movePenTo 779
SWFShape->movePen 779
SWFShape->drawLineTo 780
SWFShape->drawLine 780
SWFShape->drawCurveTo 781
SWFShape->drawCurve 781
SWFGradient 781
SWFGradient->addEntry 783
SWFBitmap 783
SWFBitmap->getWidth 785
SWFBitmap->getHeight 785
SWFFill 786
SWFFill->moveTo 786
SWFFill->scaleTo 786
SWFFill->rotateTo 787

SWFFill->skewXTo ... 787
SWFFill->skewYTo ... 787
SWFMorph ... 788
SWFMorph->getshape1 ... 789
SWFMorph->getshape2 ... 789
SWFText ... 790
SWFText->setFont ... 791
SWFText->setHeight ... 791
SWFText->setSpacing ... 791
SWFText->setColor ... 792
SWFText->moveTo ... 792
SWFText->addString ... 792
SWFText->getWidth ... 793
SWFFont ... 793
swffont->getwidth ... 794
SWFTextField ... 794
SWFTextField->setFont ... 795
SWFTextField->setbounds ... 795
SWFTextField->align ... 796
SWFTextField->setHeight ... 796
SWFTextField->setLeftMargin ... 796
SWFTextField->setrightMargin ... 797
SWFTextField->setMargins ... 797
SWFTextField->setindentation ... 797
SWFTextField->setLineSpacing ... 798
SWFTextField->setcolor ... 798
SWFTextField->setname ... 798
SWFTextField->addstring ... 799
SWFSprite ... 799
SWFSprite->add ... 800
SWFSprite->remove ... 801
SWFSprite->setframes ... 801
SWFSprite->nextframe ... 801
SWFbutton ... 802
SWFbutton->addShape ... 805
SWFbutton->setUp ... 805
SWFbutton->setOver ... 805
SWFbutton->setdown ... 806
SWFbutton->setHit ... 806
SWFbutton->addAction ... 806
SWFbutton->setAction ... 807
SWFAction ... 807
LVIII. Sonstige Funktionen ... 818
connection_aborted ... 819
connection_status ... 819
connection_timeout ... 819
define ... 819
defined ... 820
die ... 820
eval ... 820
exit ... 821
get_browser ... 822

highlight_string ... 823
highlight_file ... 823
ignore_user_abort ... 824
iptcparse ... 824
leak ... 825
pack ... 825
show_source ... 826
sleep ... 827
uniqid ... 827
unpack ... 827
usleep ... 828
LIX. mnoGoSearch Funktionen ... 829
udm_add_search_limit ... 830
udm_alloc_agent ... 830
udm_api_version ... 831
udm_cat_path ... 831
udm_cat_list ... 832
udm_clear_search_limits ... 833
udm_errno ... 833
udm_error ... 834
udm_find ... 834
udm_free_agent ... 834
udm_free_ispell_data ... 835
udm_free_res ... 835
udm_get_doc_count ... 835
udm_get_res_field ... 836
udm_get_res_param ... 836
udm_load_ispell_data ... 837
udm_set_agent_param ... 839
udm_check_charset ... 841
udm_check_stored ... 842
udm_close_stored ... 842
udm_crc32 ... 842
udm_open_stored ... 843
LX. mSQL Funktionen ... 844
msql ... 845
msql_affected_rows ... 845
msql_close ... 845
msql_connect ... 845
msql_create_db ... 846
msql_createdb ... 846
msql_data_seek ... 846
msql_dbname ... 846
msql_drop_db ... 847
msql_dropdb ... 847
msql_error ... 847
msql_fetch_array ... 847
msql_fetch_field ... 848
msql_fetch_object ... 848
msql_fetch_row ... 849
msql_fieldname ... 849
msql_field_seek ... 849

msql_fieldtable ... 849
msql_fieldtype ... 850
msql_fieldflags ... 850
msql_fieldlen ... 850
msql_free_result ... 850
msql_freeresult ... 850
msql_list_fields ... 851
msql_listfields ... 851
msql_list_dbs ... 851
msql_listdbs ... 851
msql_list_tables ... 851
msql_listtables ... 852
msql_num_fields ... 852
msql_num_rows ... 852
msql_numfields ... 852
msql_numrows ... 852
msql_pconnect ... 853
msql_query ... 853
msql_regcase ... 853
msql_result ... 854
msql_select_db ... 854
msql_selectdb ... 854
msql_tablename ... 854
LXI. MySQL Funktionen ... 856
mysql_affected_rows ... 858
mysql_change_user ... 858
mysql_close ... 858
mysql_connect ... 859
mysql_create_db ... 860
mysql_data_seek ... 861
mysql_db_name ... 861
mysql_db_query ... 862
mysql_drop_db ... 862
mysql_errno ... 863
mysql_error ... 863
mysql_escape_string ... 864
mysql_fetch_array ... 864
mysql_fetch_assoc ... 865
mysql_fetch_field ... 866
mysql_fetch_lengths ... 867
mysql_fetch_object ... 868
mysql_fetch_row ... 868
mysql_field_flags ... 869
mysql_field_name ... 869
mysql_field_len ... 870
mysql_field_seek ... 870
mysql_field_table ... 870
mysql_field_type ... 871
mysql_free_result ... 872
mysql_insert_id ... 872
mysql_list_dbs ... 872
mysql_list_fields ... 873

mysql_list_tables ... 874
mysql_num_fields ... 874
mysql_num_rows ... 875
mysql_pconnect ... 875
mysql_query ... 876
mysql_unbuffered_query ... 877
mysql_result ... 878
mysql_select_db ... 878
mysql_tablename ... 879
mysql_get_client_info ... 879
mysql_get_host_info ... 879
mysql_get_proto_info ... 880
mysql_get_server_info ... 880
LXII. Mohawk Software Session Handler Funktionen ... 881
msession_connect ... 882
msession_disconnect ... 882
msession_count ... 882
msession_create ... 882
msession_destroy ... 883
msession_lock ... 883
msession_unlock ... 883
msession_set ... 883
msession_get ... 884
msession_uniq ... 884
msession_randstr ... 884
msession_find ... 884
msession_list ... 885
msession_get_array ... 885
msession_set_array ... 885
msession_listvar ... 885
msession_timeout ... 885
msession_inc ... 886
msession_getdata ... 886
msession_setdata ... 886
msession_plugin ... 887
LXIII. muscat functions ... 888
muscat_setup ... 889
muscat_setup_net ... 889
muscat_give ... 889
muscat_get ... 890
muscat_close ... 890
LXIV. Netzwerk Funktionen ... 892
checkdnsrr ... 893
closelog ... 893
debugger_off ... 893
debugger_on ... 893
define_syslog_variables ... 893
fsockopen ... 894
gethostbyaddr ... 895
gethostbyname ... 895
gethostbynamel ... 895
getmxrr ... 896

getprotobyname ... 896
getprotobynumber ... 896
getservbyname ... 896
getservbyport ... 897
ip2long ... 897
long2ip ... 897
openlog ... 898
pfsockopen ... 899
socket_get_status ... 899
socket_set_blocking ... 899
socket_set_timeout ... 900
syslog ... 900
LXV. Ncurses terminal screen control functions ... 902
ncurses_can_change_color ... 907
ncurses_cbreak ... 907
ncurses_clear ... 907
ncurses_clrtobot ... 908
ncurses_clrtoeol ... 908
ncurses_def_prog_mode ... 908
ncurses_def_shell_mode ... 909
ncurses_delch ... 909
ncurses_deleteln ... 910
ncurses_doupdate ... 910
ncurses_echo ... 910
ncurses_erase ... 911
ncurses_erasechar ... 911
ncurses_flash ... 911
ncurses_flushinp ... 912
ncurses_has_colors ... 912
ncurses_has_ic ... 913
ncurses_has_il ... 913
ncurses_inch ... 913
ncurses_insertln ... 914
ncurses_isendwin ... 914
ncurses_killchar ... 914
ncurses_nl ... 915
ncurses_nocbreak ... 915
ncurses_noecho ... 915
ncurses_nonl ... 916
ncurses_noraw ... 916
ncurses_raw ... 917
ncurses_resetty ... 917
ncurses_savetty ... 917
ncurses_slk_init ... 918
ncurses_slk_attr ... 918
ncurses_slk_clear ... 918
ncurses_slk_noutrefresh ... 919
ncurses_slk_refresh ... 919
ncurses_slk_restore ... 919
ncurses_slk_touch ... 920
ncurses_termattrs ... 920
ncurses_use_default_colors ... 920

ncurses_addch....921
ncurses_addchnstr....921
ncurses_addchstr....921
ncurses_addnstr....922
ncurses_addstr....922
ncurses_assume_default_colors....922
ncurses_attroff....923
ncurses_attron....923
ncurses_attrset....923
ncurses_baudrate....924
ncurses_beep....924
ncurses_bkgd....924
ncurses_border....925
ncurses_color_set....925
ncurses_curs_set....925
ncurses_define_key....926
ncurses_delay_output....926
ncurses_delwin....926
ncurses_echochar....927
ncurses_end....927
ncurses_filter....927
ncurses_getch....928
ncurses_halfdelay....928
ncurses_has_key....928
ncurses_hline....929
ncurses_init....929
ncurses_init_color....929
ncurses_init_pair....930
ncurses_insch....930
ncurses_insdelln....930
ncurses_insstr....931
ncurses_instr....931
ncurses_keyok....931
ncurses_mouseinterval....932
ncurses_move....932
ncurses_mvaddch....932
ncurses_mvaddchnstr....933
ncurses_mvaddchstr....933
ncurses_mvaddnstr....933
ncurses_mvaddstr....934
ncurses_mvcur....934
ncurses_mvdelch....934
ncurses_mvgetch....935
ncurses_mvhline....935
ncurses_mvinch....935
ncurses_mvvline....936
ncurses_mvwaddstr....936
ncurses_napms....936
ncurses_newwin....937
ncurses_noqiflush....937
ncurses_putp....937
ncurses_qiflush....938

ncurses_refresh ... 938
ncurses_scr_dump ... 938
ncurses_scr_init ... 939
ncurses_scr_restore ... 939
ncurses_scr_set ... 939
ncurses_scrl ... 940
ncurses_slk_attroff ... 940
ncurses_slk_attron ... 940
ncurses_slk_attrset ... 941
ncurses_slk_color ... 941
ncurses_standend ... 941
ncurses_standout ... 942
ncurses_start_color ... 942
ncurses_typeahead ... 942
ncurses_ungetch ... 943
ncurses_use_extended_names ... 943
ncurses_vidattr ... 943
ncurses_vline ... 944
ncurses_wrefresh ... 944
ncurses_bkgdset ... 944
ncurses_timeout ... 945
ncurses_use_env ... 945
ncurses_termname ... 945
ncurses_longname ... 946
ncurses_mousemask ... 946
ncurses_getmouse ... 948
ncurses_ungetmouse ... 949
LXVI. Lotus Notes functions ... 950
notes_create_db ... 951
notes_drop_db ... 951
notes_version ... 951
notes_create_note ... 952
notes_mark_read ... 952
notes_mark_unread ... 953
notes_unread ... 953
notes_header_info ... 954
notes_body ... 954
notes_find_note ... 955
notes_nav_create ... 955
notes_search ... 956
notes_copy_db ... 956
notes_list_msgs ... 957
LXVII. ODBC Funktionen ... 958
odbc_autocommit ... 959
odbc_binmode ... 959
odbc_close ... 960
odbc_close_all ... 960
odbc_commit ... 960
odbc_connect ... 961
odbc_cursor ... 961
odbc_do ... 961
odbc_exec ... 962

odbc_execute ... 962
odbc_fetch_into ... 962
odbc_fetch_row ... 962
odbc_field_name ... 963
odbc_field_type ... 963
odbc_field_len ... 963
odbc_free_result ... 963
odbc_longreadlen ... 964
odbc_num_fields ... 964
odbc_pconnect ... 964
odbc_prepare ... 965
odbc_num_rows ... 965
odbc_result ... 965
odbc_result_all ... 966
odbc_rollback ... 966
odbc_setoption ... 966
LXVIII. Oracle 8 Funktionen ... 968
OCIDefineByName ... 970
OCIBindByName ... 970
OCILogon ... 972
OCIPLogon ... 973
OCINLogon ... 974
OCILogOff ... 976
OCIExecute ... 976
OCICommit ... 976
OCIRollback ... 976
OCINewDescriptor ... 976
OCIRowCount ... 978
OCINumCols ... 978
OCIResult ... 979
OCIFetch ... 979
OCIFetchInto ... 979
OCIFetchStatement ... 980
OCIColumnIsNULL ... 981
OCIColumnSize ... 981
OCIServerVersion ... 982
OCIStatementType ... 982
OCINewCursor ... 983
OCIFreeStatement ... 984
OCIFreeCursor ... 984
OCIColumnName ... 985
OCIColumnType ... 985
OCIParse ... 986
OCIError ... 986
OCIInternalDebug ... 987
OCICancel ... 987
OCISetPrefetch ... 987
OCIWriteLobToFile ... 987
OCISaveLobFile ... 987
OCISaveLob ... 988
OCILoadLob ... 988
OCIColumnScale ... 988

OCIColumnPrecision ... 988
OCIColumnTypeRaw ... 988
OCINewCollection ... 989
OCIFreeCollection ... 989
OCICollAssign ... 989
OCICollAssignElem ... 989
OCICollGetElem ... 989
OCICollMax ... 989
OCICollSize ... 990
OCICollTrim ... 990
LXIX. OpenSSL Funktionen ... 991
openssl_error_string ... 995
openssl_free_key ... 995
openssl_get_privatekey ... 996
openssl_get_publickey ... 996
openssl_open ... 996
openssl_seal ... 997
openssl_sign ... 998
openssl_verify ... 999
openssl_pkcs7_decrypt ... 1000
openssl_pkcs7_encrypt ... 1001
openssl_pkcs7_sign ... 1002
openssl_pkcs7_verify ... 1004
openssl_x509_checkpurpose ... 1004
openssl_x509_free ... 1005
openssl_x509_parse ... 1006
openssl_x509_read ... 1006
LXX. Oracle Funktionen ... 1008
Ora_Bind ... 1009
Ora_Close ... 1009
Ora_ColumnName ... 1009
Ora_ColumnSize ... 1010
Ora_ColumnType ... 1010
Ora_Commit ... 1010
Ora_CommitOff ... 1010
Ora_CommitOn ... 1011
Ora_Do ... 1011
Ora_Error ... 1011
Ora_ErrorCode ... 1012
Ora_Exec ... 1012
Ora_Fetch ... 1012
Ora_Fetch_Into ... 1012
Ora_GetColumn ... 1013
Ora_Logoff ... 1013
Ora_Logon ... 1013
Ora_pLogon ... 1014
Ora_Numcols ... 1014
Ora_Numrows ... 1014
Ora_Open ... 1014
Ora_Parse ... 1015
Ora_Rollback ... 1015
LXXI. Ovrimos SQL functions ... 1016

ovrimos_connect ... 1017
ovrimos_close ... 1017
ovrimos_longreadlen ... 1017
ovrimos_prepare ... 1018
ovrimos_execute ... 1018
ovrimos_cursor ... 1019
ovrimos_exec ... 1019
ovrimos_fetch_into ... 1019
ovrimos_fetch_row ... 1020
ovrimos_result ... 1021
ovrimos_result_all ... 1021
ovrimos_num_rows ... 1023
ovrimos_num_fields ... 1023
ovrimos_field_name ... 1023
ovrimos_field_type ... 1023
ovrimos_field_len ... 1024
ovrimos_field_num ... 1024
ovrimos_free_result ... 1024
ovrimos_commit ... 1024
ovrimos_rollback ... 1024
LXXII. Funktionen zur Ausgabesteuerung ... 1026
flush ... 1027
ob_start ... 1027
ob_get_contents ... 1027
ob_end_flush ... 1027
ob_end_clean ... 1027
ob_implicit_flush ... 1028
LXXIII. Object property and method call overloading ... 1029
overload ... 1031
LXXIV. PDF Funktionen ... 1032
pdf_set_info ... 1038
pdf_open ... 1038
pdf_close ... 1039
pdf_begin_page ... 1039
pdf_end_page ... 1039
pdf_show ... 1039
pdf_show_boxed ... 1039
pdf_show_xy ... 1040
pdf_set_font ... 1040
pdf_set_leading ... 1041
pdf_set_parameter ... 1041
pdf_get_parameter ... 1041
pdf_set_value ... 1041
pdf_get_value ... 1041
pdf_get_image_height ... 1042
pdf_get_image_width ... 1042
pdf_set_text_rendering ... 1042
pdf_set_horiz_scaling ... 1042
pdf_set_text_rise ... 1043
pdf_set_text_matrix ... 1043
pdf_set_text_pos ... 1043
pdf_set_char_spacing ... 1043

pdf_set_word_spacing 1043
pdf_skew 1044
pdf_continue_text 1044
pdf_stringwidth 1044
pdf_save 1044
pdf_restore 1045
pdf_translate 1045
pdf_scale 1045
pdf_rotate 1046
pdf_setflat 1046
pdf_setlinejoin 1046
pdf_setlinecap 1046
pdf_setmiterlimit 1047
pdf_setlinewidth 1047
pdf_setdash 1047
pdf_moveto 1047
pdf_curveto 1047
pdf_lineto 1048
pdf_circle 1048
pdf_arc 1048
pdf_rect 1048
pdf_closepath 1049
pdf_stroke 1049
pdf_closepath_stroke 1049
pdf_fill 1049
pdf_fill_stroke 1050
pdf_closepath_fill_stroke 1050
pdf_endpath 1050
pdf_clip 1050
pdf_setgray_fill 1051
pdf_setgray_stroke 1051
pdf_setgray 1051
pdf_setrgbcolor_fill 1051
pdf_setrgbcolor_stroke 1051
pdf_setrgbcolor 1052
pdf_add_outline 1052
pdf_set_transition 1052
pdf_set_duration 1052
pdf_open_gif 1053
pdf_open_png 1053
pdf_open_image_file 1054
pdf_open_memory_image 1054
pdf_open_jpeg 1055
pdf_open_tiff 1055
pdf_close_image 1055
pdf_place_image 1055
pdf_put_image 1056
pdf_execute_image 1056
pdf_add_annotation 1057
pdf_set_border_style 1057
pdf_set_border_color 1057
pdf_set_border_dash 1057

LXXV. Verisign Payflow Pro Funktionen ... 1059
pfpro_init ... 1060
pfpro_cleanup ... 1060
pfpro_process ... 1060
pfpro_process_raw ... 1061
pfpro_version ... 1062
LXXVI. PHP Optionen und Informationen ... 1063
assert ... 1064
assert_options ... 1064
dl ... 1065
extension_loaded ... 1065
getenv ... 1066
get_cfg_var ... 1066
get_current_user ... 1066
get_magic_quotes_gpc ... 1066
get_magic_quotes_runtime ... 1067
getlastmod ... 1067
getmyinode ... 1067
getmypid ... 1067
getmyuid ... 1068
getrusage ... 1068
phpinfo ... 1068
php_logo_guid ... 1069
php_sapi_name ... 1069
phpversion ... 1069
putenv ... 1070
set_magic_quotes_runtime ... 1070
set_time_limit ... 1070
zend_logo_guid ... 1071
get_loaded_extensions ... 1071
get_extension_funcs ... 1072
get_required_files ... 1072
get_included_files ... 1073
LXXVII. POSIX Funktionen ... 1074
posix_kill ... 1075
posix_getpid ... 1075
posix_getppid ... 1075
posix_getuid ... 1075
posix_geteuid ... 1075
posix_getgid ... 1076
posix_getegid ... 1076
posix_setuid ... 1076
posix_seteuid ... 1076
posix_setgid ... 1077
posix_setegid ... 1077
posix_getgroups ... 1077
posix_getlogin ... 1077
posix_getpgrp ... 1078
posix_setsid ... 1078
posix_setpgid ... 1078
posix_getpgid ... 1078
posix_getsid ... 1079

posix_uname ... 1079
posix_times ... 1079
posix_ctermid ... 1080
posix_ttyname ... 1080
posix_isatty ... 1080
posix_getcwd ... 1080
posix_mkfifo ... 1080
posix_getgrnam ... 1081
posix_getgrgid ... 1081
posix_getpwnam ... 1081
posix_getpwuid ... 1082
posix_getrlimit ... 1083
LXXVIII. PostgreSQL Funktionen ... 1084
pg_Close ... 1086
pg_cmdTuples ... 1086
pg_Connect ... 1086
pg_DBname ... 1087
pg_end_copy ... 1087
pg_ErrorMessage ... 1087
pg_Exec ... 1087
pg_Fetch_Array ... 1088
pg_Fetch_Object ... 1089
pg_Fetch_Row ... 1090
pg_FieldIsNull ... 1091
pg_FieldName ... 1091
pg_FieldNum ... 1091
pg_FieldPrtLen ... 1091
pg_FieldSize ... 1092
pg_FieldType ... 1092
pg_FreeResult ... 1092
pg_GetLastOid ... 1092
pg_Host ... 1093
pg_loclose ... 1093
pg_locreate ... 1093
pg_loexport ... 1093
pg_loimport ... 1094
pg_loopen ... 1094
pg_loread ... 1094
pg_loreadall ... 1094
pg_lounlink ... 1094
pg_lowrite ... 1095
pg_NumFields ... 1095
pg_NumRows ... 1095
pg_Options ... 1095
pg_pConnect ... 1096
pg_Port ... 1096
pg_put_line ... 1096
pg_Result ... 1097
pg_set_client_encoding ... 1097
pg_client_encoding ... 1097
pg_trace ... 1098
pg_tty ... 1098

pg_untrace 1098
LXXIX. Process Control Functions 1100
pcntl_fork 1102
pcntl_signal 1102
pcntl_waitpid 1103
pcntl_wexitstatus 1104
pcntl_wifexited 1104
pcntl_wifsignaled 1105
pcntl_wifstopped 1105
pcntl_wstopsig 1105
pcntl_wtermsig 1105
pcntl_exec 1106
LXXX. Funktionen zur Programmausführung 1107
escapeshellarg 1108
escapeshellcmd 1108
exec 1108
passthru 1109
system 1109
LXXXI. Printer functions 1111
printer_open 1112
printer_abort 1112
printer_close 1112
printer_write 1113
printer_list 1113
printer_set_option 1114
printer_get_option 1115
printer_create_dc 1116
printer_delete_dc 1116
printer_start_doc 1117
printer_end_doc 1117
printer_start_page 1117
printer_end_page 1117
printer_create_pen 1118
printer_delete_pen 1118
printer_select_pen 1118
printer_create_brush 1119
printer_delete_brush 1119
printer_select_brush 1120
printer_create_font 1120
printer_delete_font 1121
printer_select_font 1121
printer_logical_fontheight 1122
printer_draw_roundrect 1122
printer_draw_rectangle 1123
printer_draw_elipse 1124
printer_draw_text 1124
printer_draw_line 1125
printer_draw_chord 1126
printer_draw_pie 1126
printer_draw_bmp 1127
LXXXII. Pspell Functions 1129
pspell_add_to_personal 1130

pspell_add_to_session ... 1130
pspell_check ... 1130
pspell_clear_session ... 1131
pspell_config_create ... 1131
pspell_config_ignore ... 1132
pspell_config_mode ... 1132
pspell_config_personal ... 1133
pspell_config_repl ... 1134
pspell_config_runtogether ... 1134
pspell_config_save_repl ... 1135
pspell_new ... 1135
pspell_new_config ... 1136
pspell_new_personal ... 1136
pspell_save_wordlist ... 1137
pspell_store_replacement ... 1138
pspell_suggest ... 1138
LXXXIII. GNU Readline ... 1140
readline ... 1141
readline_add_history ... 1141
readline_clear_history ... 1141
readline_completion_function ... 1141
readline_info ... 1142
readline_list_history ... 1142
readline_read_history ... 1142
readline_write_history ... 1142
LXXXIV. GNU Recode Funktionen ... 1143
recode_string ... 1144
recode ... 1144
recode_file ... 1144
LXXXV. Regular Expression Functions (Perl-Compatible) ... 1146
preg_match ... 1147
preg_match_all ... 1148
preg_replace ... 1150
preg_replace_callback ... 1152
preg_split ... 1152
preg_quote ... 1153
preg_grep ... 1154
Pattern Modifiers ... 1154
Pattern Syntax ... 1156
LXXXVI. qtdom functions ... 1181
qdom_tree ... 1182
qdom_error ... 1182
LXXXVII. Reguläre Ausdrücke Funktionen (POSIX erweitert) ... 1183
ereg ... 1185
ereg_replace ... 1185
eregi ... 1187
eregi_replace ... 1187
split ... 1187
spliti ... 1189
sql_regcase ... 1189
LXXXVIII. Semaphor und Shared Memory Funktionen ... 1190
sem_get ... 1192

sem_acquire ... 1192
sem_release ... 1192
shm_attach ... 1193
shm_detach ... 1193
shm_remove ... 1193
shm_put_var ... 1194
shm_get_var ... 1194
shm_remove_var ... 1194
LXXXIX. SESAM Datenbankfunktionen ... 1195
sesam_connect ... 1200
sesam_disconnect ... 1200
sesam_settransaction ... 1201
sesam_commit ... 1202
sesam_rollback ... 1202
sesam_execimm ... 1203
sesam_query ... 1203
sesam_num_fields ... 1205
sesam_field_name ... 1205
sesam_diagnostic ... 1206
sesam_fetch_result ... 1207
sesam_affected_rows ... 1208
sesam_errormsg ... 1209
sesam_field_array ... 1209
sesam_fetch_row ... 1212
sesam_fetch_array ... 1214
sesam_seek_row ... 1215
sesam_free_result ... 1216
XC. Session Funktionen ... 1217
session_start ... 1223
session_destroy ... 1223
session_name ... 1224
session_module_name ... 1224
session_save_path ... 1225
session_id ... 1225
session_register ... 1225
session_unregister ... 1226
session_unset ... 1227
session_is_registered ... 1227
session_get_cookie_params ... 1228
session_set_cookie_params ... 1228
session_decode ... 1228
session_encode ... 1228
session_set_save_handler ... 1229
session_cache_limiter ... 1230
session_cache_expire ... 1231
session_write_close ... 1231
XCI. Shared Memory Funktionen (Gemeinsamer Speicher) ... 1233
shmop_open ... 1234
shmop_read ... 1234
shmop_write ... 1235
shmop_size ... 1235
shmop_delete ... 1236

shmop_close 1236
XCII. Shockwave Flash Funktionen 1238
swf_openfile 1240
swf_closefile 1240
swf_labelframe 1240
swf_showframe 1240
swf_setframe 1240
swf_getframe 1241
swf_mulcolor 1241
swf_addcolor 1241
swf_placeobject 1241
swf_modifyobject 1242
swf_removeobject 1242
swf_nextid 1242
swf_startdoaction 1242
swf_actiongotoframe 1242
swf_actiongeturl 1243
swf_actionnextframe 1243
swf_actionprevframe 1243
swf_actionplay 1243
swf_actionstop 1243
swf_actiontogglequality 1244
swf_actionwaitforframe 1244
swf_actionsettarget 1244
swf_actiongotolabel 1244
swf_enddoaction 1244
swf_defineline 1245
swf_definerect 1245
swf_definepoly 1245
swf_startshape 1245
swf_shapelinesolid 1245
swf_shapefilloff 1246
swf_shapefillsolid 1246
swf_shapefillbitmapclip 1246
swf_shapefillbitmaptile 1246
swf_shapemoveto 1246
swf_shapelineto 1247
swf_shapecurveto 1247
swf_shapecurveto3 1247
swf_shapearc 1247
swf_endshape 1248
swf_definefont 1248
swf_setfont 1248
swf_fontsize 1248
swf_fontslant 1248
swf_fonttracking 1249
swf_getfontinfo 1249
swf_definetext 1249
swf_textwidth 1249
swf_definebitmap 1249
swf_getbitmapinfo 1250
swf_startsymbol 1250

swf_endsymbol ... 1250
swf_startbutton ... 1250
swf_addbuttonrecord ... 1251
swf_oncondition ... 1251
swf_endbutton ... 1252
swf_viewport ... 1252
swf_ortho ... 1252
swf_ortho2 ... 1252
swf_perspective ... 1253
swf_polarview ... 1253
swf_lookat ... 1253
swf_pushmatrix ... 1254
swf_popmatrix ... 1254
swf_scale ... 1254
swf_translate ... 1254
swf_rotate ... 1254
swf_posround ... 1255
XCIII. SNMP Funktionen ... 1256
snmpget ... 1257
snmpset ... 1257
snmpwalk ... 1257
snmpwalkoid ... 1258
snmp_get_quick_print ... 1258
snmp_set_quick_print ... 1259
XCIV. Socket Funktionen ... 1261
accept_connect ... 1264
bind ... 1264
close ... 1265
connect ... 1265
listen ... 1266
read ... 1266
socket ... 1267
strerror ... 1267
write ... 1268
XCV. String-Funktionen ... 1270
AddCSlashes ... 1271
AddSlashes ... 1271
bin2hex ... 1271
chop ... 1271
chr ... 1272
chunk_split ... 1272
convert_cyr_string ... 1273
count_chars ... 1273
crc32 ... 1274
crypt ... 1274
echo ... 1275
explode ... 1275
get_html_translation_table ... 1276
get_meta_tags ... 1276
hebrev ... 1277
hebrevc ... 1277
htmlentities ... 1277

htmlspecialchars 1278
implode 1279
join 1279
levenshtein 1279
ltrim 1280
md5 1280
metaphone 1280
nl2br 1280
ord 1281
parse_str 1281
print 1281
printf 1282
quoted_printable_decode 1282
QuoteMeta 1282
rtrim 1282
sscanf 1283
setlocale 1284
similar_text 1284
soundex 1284
sprintf 1285
strcasecmp 1286
strchr 1287
strcmp 1287
strcspn 1287
strip_tags 1287
StripCSlashes 1288
StripSlashes 1288
stristr 1288
strlen 1289
strnatcmp 1289
strnatcasecmp 1290
strncmp 1290
str_pad 1290
strpos 1291
strrchr 1292
str_repeat 1292
strrev 1293
strrpos 1293
strspn 1293
strstr 1294
strtok 1294
strtolower 1295
strtoupper 1295
str_replace 1295
strtr 1296
substr 1297
substr_count 1297
substr_replace 1298
trim 1299
ucfirst 1299
ucwords 1299
wordwrap 1300

XCVI. Sybase functions ... 1302
sybase_affected_rows ... 1303
sybase_close ... 1303
sybase_connect ... 1303
sybase_data_seek ... 1304
sybase_fetch_array ... 1304
sybase_fetch_field ... 1304
sybase_fetch_object ... 1305
sybase_fetch_row ... 1305
sybase_field_seek ... 1305
sybase_free_result ... 1306
sybase_get_last_message ... 1306
sybase_min_client_severity ... 1306
sybase_min_error_severity ... 1306
sybase_min_message_severity ... 1307
sybase_min_server_severity ... 1307
sybase_num_fields ... 1307
sybase_num_rows ... 1307
sybase_pconnect ... 1307
sybase_query ... 1308
sybase_result ... 1308
sybase_select_db ... 1308
XCVII. URL Funktionen ... 1310
base64_decode ... 1311
base64_encode ... 1311
parse_url ... 1311
rawurldecode ... 1311
rawurlencode ... 1312
urldecode ... 1312
urlencode ... 1313
XCVIII. Variablen-Funktionen ... 1314
doubleval ... 1315
empty ... 1315
gettype ... 1315
intval ... 1316
is_array ... 1316
is_bool ... 1316
is_double ... 1316
is_float ... 1317
is_int ... 1317
is_integer ... 1317
is_long ... 1317
is_numeric ... 1317
is_object ... 1318
is_real ... 1318
is_resource ... 1318
is_string ... 1318
isset ... 1319
print_r ... 1319
serialize ... 1320
settype ... 1320
strval ... 1321

unserialize ... 1321
unset ... 1322
var_dump ... 1322
XCIX. vpopmail functions ... 1323
vpopmail_add_domain ... 1324
vpopmail_del_domain ... 1324
vpopmail_add_alias_domain ... 1324
vpopmail_add_domain_ex ... 1325
vpopmail_del_domain_ex ... 1325
vpopmail_add_alias_domain_ex ... 1326
vpopmail_add_user ... 1326
vpopmail_del_user ... 1327
vpopmail_passwd ... 1327
vpopmail_set_user_quota ... 1328
vpopmail_auth_user ... 1328
vpopmail_alias_add ... 1329
vpopmail_alias_del ... 1329
vpopmail_alias_del_domain ... 1330
vpopmail_alias_get ... 1330
vpopmail_alias_get_all ... 1331
vpopmail_error ... 1331
C. W32api functions ... 1333
w32api_set_call_method ... 1334
w32api_register_function ... 1334
w32api_invoke_function ... 1334
w32api_deftype ... 1335
w32api_init_dtype ... 1335
CI. WDDX Funktionen ... 1337
wddx_serialize_value ... 1339
wddx_serialize_vars ... 1339
wddx_packet_start ... 1339
wddx_packet_end ... 1340
wddx_add_vars ... 1340
wddx_deserialize ... 1340
CII. XML parser functions ... 1341
xml_parser_create ... 1350
xml_set_object ... 1350
xml_set_element_handler ... 1351
xml_set_character_data_handler ... 1352
xml_set_processing_instruction_handler ... 1352
xml_set_default_handler ... 1353
xml_set_unparsed_entity_decl_handler ... 1354
xml_set_notation_decl_handler ... 1355
xml_set_external_entity_ref_handler ... 1355
xml_parse ... 1356
xml_get_error_code ... 1357
xml_error_string ... 1357
xml_get_current_line_number ... 1357
xml_get_current_column_number ... 1358
xml_get_current_byte_index ... 1358
xml_parse_into_struct ... 1358
xml_parser_free ... 1362

xml_parser_set_option 1362
xml_parser_get_option 1363
utf8_decode 1363
utf8_encode 1364
xml_parser_create_ns 1364
xml_set_end_namespace_decl_handler 1364
xml_set_start_namespace_decl_handler 1365
CIII. XMLRPC functions 1366
xmlrpc_encode_request 1367
xmlrpc_encode 1367
xmlrpc_decode_request 1367
xmlrpc_decode 1368
xmlrpc_server_create 1368
xmlrpc_server_destroy 1369
xmlrpc_server_register_method 1369
xmlrpc_server_register_introspection_callback 1370
xmlrpc_server_call_method 1370
xmlrpc_server_add_introspection_data 1371
xmlrpc_parse_method_descriptions 1371
xmlrpc_set_type 1372
xmlrpc_get_type 1372
CIV. XSLT Funktionen 1374
xslt_closelog 1375
xslt_create 1375
xslt_errno 1375
xslt_error 1375
xslt_fetch_result 1375
xslt_free 1376
xslt_openlog 1376
xslt_output_begintransform 1376
xslt_output_endtransform 1376
xslt_output_process 1376
xslt_run 1377
xslt_set_sax_handler 1377
xslt_transform 1377
CV. YAZ functions 1378
yaz_addinfo 1380
yaz_close 1380
yaz_connect 1380
yaz_errno 1381
yaz_error 1381
yaz_hits 1381
yaz_element 1381
yaz_database 1382
yaz_range 1382
yaz_record 1382
yaz_search 1382
yaz_present 1384
yaz_syntax 1384
yaz_scan 1384
yaz_scan_result 1385
yaz_ccl_conf 1385

yaz_ccl_parse ... 1386
yaz_itemorder ... 1386
yaz_wait ... 1388
yaz_sort ... 1389
CVI. YP/NIS Funktionen ... 1390
yp_get_default_domain ... 1391
yp_order ... 1391
yp_master ... 1391
yp_match ... 1392
yp_first ... 1392
yp_next ... 1393
CVII. ZIP Funktionen (Lesezugriff) ... 1394
zip_open ... 1396
zip_read ... 1396
zip_close ... 1396
zip_entry_close ... 1396
zip_entry_compressedsize ... 1397
zip_entry_compressionmethod ... 1397
zip_entry_filesize ... 1397
zip_entry_name ... 1397
zip_entry_open ... 1398
zip_entry_read ... 1398
CVIII. Zlib Compression Functions ... 1399
gzclose ... 1401
gzeof ... 1401
gzfile ... 1401
gzgetc ... 1401
gzgets ... 1401
gzgetss ... 1402
gzopen ... 1402
gzpassthru ... 1403
gzputs ... 1403
gzread ... 1403
gzrewind ... 1404
gzseek ... 1404
gztell ... 1404
gzwrite ... 1404
readgzfile ... 1405
gzcompress ... 1405
gzuncompress ... 1406
gzdeflate ... 1406
gzinflate ... 1406
gzencode ... 1406
V. Extending PHP 4.0 ... 1408
24. Overview ... 1408
What Is Zend? and What Is PHP? ... 1409
25. Extension Possibilities ... 1410
External Modules ... 1411
Built-in Modules ... 1411
The Zend Engine ... 1412
26. Source Layout ... 1413

Extension Conventions ... 1415
Macros ... 1415
Memory Management ... 1415
Directory and File Functions ... 1416
String Handling ... 1416
Complex Types ... 1416
27. PHP's Automatic Build System ... 1417
28. Creating Extensions ... 1420
Compiling Modules ... 1422
29. Using Extensions ... 1424
30. Troubleshooting ... 1426
31. Source Discussion ... 1428
Module Structure ... 1429
Header File Inclusions ... 1429
Declaring Exported Functions ... 1429
Declaration of the Zend Function Block ... 1430
Declaration of the Zend Module Block ... 1431
Creation of **get_module()** ... 1433
Implementation of All Exported Functions ... 1433
Summary ... 1434
32. Accepting Arguments ... 1435
Determining the Number of Arguments ... 1436
Retrieving Arguments ... 1437
Old way of retrieving arguments (deprecated) ... 1439
Dealing with a Variable Number of Arguments/Optional Parameters ... 1440
Accessing Arguments ... 1442
Dealing with Arguments Passed by Reference ... 1445
Assuring Write Safety for Other Parameters ... 1447
33. Creating Variables ... 1448
Overview ... 1449
Longs (Integers) ... 1451
Doubles (Floats) ... 1452
Strings ... 1452
Booleans ... 1453
Arrays ... 1453
Objects ... 1456
Resources ... 1457
Macros for Automatic Global Variable Creation ... 1461
Creating Constants ... 1462
34. Duplicating Variable Contents: The Copy Constructor ... 1463
35. Returning Values ... 1465
36. Printing Information ... 1468
zend_printf() ... 1469
zend_error() ... 1469
Including Output in phpinfo() ... 1469
Execution Information ... 1470
37. Startup and Shutdown Functions ... 1472
38. Calling User Functions ... 1474
39. Initialization File Support ... 1477
40. Where to Go from Here ... 1480
41. Reference: Some Configuration Macros ... 1482
`config.m4` ... 1483

42. API Macros 1484
VI. FAQ: Frequently Asked Questions 1486
43. General Information 1486
44. Mailing lists 1489
45. Obtaining PHP 1492
46. Database issues 1495
47. Installation ..... ??
48. Build Problems ..... ??
49. Using PHP ..... ??
50. PHP and HTML ..... ??
51. PHP and COM ..... ??
52. PHP und andere Sprachen ..... ??
53. Von PHP 2 auf PHP 3 umsteigen ..... ??
54. Von PHP 3 auf PHP 4 umsteigen ..... ??
55. Verschiedene Fragen ..... ??
VII. Anhang ..... ??
A. Die Geschichte von PHP und verwandten Projekten ..... ??
Die Geschichte von PHP ..... ??
PHP/FI ..... ??
PHP 3 ..... ??
PHP 4 ..... ??
Geschichte von PHP verwandten Projekten ..... ??
PEAR ..... ??
PHP Initiative zur Qualitätssicherung ..... ??
PHP-GTK ..... ??
Bücher über PHP ..... ??
Publikationen über PHP ..... ??
B. Using PHP from the command line ..... ??
C. Migrating from PHP 3 to PHP 4 ..... ??
What has changed in PHP 4 ..... ??
Running PHP 3 and PHP 4 concurrently ..... ??
Migrating Configuration Files ..... ??
Parser behavior ..... ??
Error reporting ..... ??
Configuration changes ..... ??
Additional warning messages ..... ??
Initializers ..... ??
`empty("0")` ..... ??
Missing functions ..... ??
Functions missing due to conceptual changes ..... ??
Deprecate functions and extensions ..... ??
Changed status for unset() ..... ??
PHP 3 extension ..... ??
Variable substitution in strings ..... ??
Cookies ..... ??
Handling of global variables ..... ??
D. Migrating von PHP/FI 2.0 zu PHP 3.0 ..... ??
Über die Inkompatbilitäten in 3.0 ..... ??
Start-/End-Tags ..... ??
if..endif Syntax ..... ??
while Syntax ..... ??

Ausdruckstypen ..... ??
Änderung der Fehlermeldungen ..... ??
Short-circuited boolean evaluation ..... ??
Function TRUE/false return values ..... ??
Andere Inkompatibilitäten ..... ??
E. Der PHP-Debugger ..... ??
Verwendung des Debuggers ..... ??
Debugger-Protokoll ..... ??
F. PHP development ..... ??
Adding functions to PHP3 ..... ??
Function Prototype ..... ??
Function Arguments ..... ??
Variable Function Arguments ..... ??
Using the Function Arguments ..... ??
Memory Management in Functions ..... ??
Setting Variables in the Symbol Table ..... ??
Returning simple values ..... ??
Returning complex values ..... ??
Using the resource list ..... ??
Using the persistent resource table ..... ??
Adding runtime configuration directives ..... ??
Calling User Functions ..... ??
HashTable *function_table ..... ??
pval *object ..... ??
pval *function_name ..... ??
pval *retval ..... ??
int param_count ..... ??
pval *params[] ..... ??
Reporting Errors ..... ??
E_NOTICE ..... ??
E_WARNING ..... ??
E_ERROR ..... ??
E_PARSE ..... ??
E_CORE_ERROR ..... ??
E_CORE_WARNING ..... ??
E_COMPILE_ERROR ..... ??
E_COMPILE_WARNING ..... ??
E_USER_ERROR ..... ??
E_USER_WARNING ..... ??
E_USER_NOTICE ..... ??
G. Liste der Funktion Aliase ..... ??
H. Reservierte Schlüsselwörter in PHP ..... ??
I. Liste von Ressourcentypen ..... ??
J. Über das Handbuch ..... ??
Formate ..... ??
Über die Kommentare (User Notes) ..... ??
Mehr Informationen über PHP finden ..... ??
Wie Sie helfen können, die Dokumentation zu verbessern ..... ??
Wir wir die Formate generieren ..... ??
K. missing stuff ..... ??

Vorwort

PHP ist die Abkürzung für "PHP: Hypertext Preprocessor", eine Skriptsprache, die sich in HTML einbinden lässt. Viele der syntaktischen Möglichkeiten sind den Programmiersprachen C, Java und Perl entnommen und es wurden auch einige PHP-spezifische Features entwickelt. Das Ziel der Sprache ist es, das Schreiben von Programmen zur Erzeugung von dynamisch generierten Seiten zu erleichtern und zu beschleunigen.

Über dieses Handbuch

Dieses Handbuch ist in XML geschrieben und verwendet die DocBook XML DTD (http://www.nwalsh.com/docbook/xml/) und DSSSL (http://www.jclark.com/dsssl/) (Document Style and Semantics Specification Language) für die Formatierung. Die Tools zur Transformation in HTML, TeX und RTF sind Jade (http://www.jclark.com/jade/) von James Clark (http://www.jclark.com/bio.htm) und die Modularen DocBook Stylesheets (http://nwalsh.com/docbook/dsssl/) von Norman Walsh (http://nwalsh.com/). Die Programme zur Herstellung des PHP Handbuchs werden von Stig Sæther Bakken (mailto:stig@php.net) gepflegt.

Dieses HTML Handbuch wird jeden Tag neu erzeugt und kann bei http://www.php.net/download-docs.php bezogen werden.

Teil I. Einführung

Kapitel 1. Einleitung

Was ist PHP?

PHP PHP (Akronym für "PHP: Hypertext Preprocessor") ist eine weit verbreitete und für den allgemeinen Gebrauch bestimmte Open Source Skriptsprache, welche speziell für die Webprogrammierung geeignet ist, und in HTML eingebettet werden kann.

Hört sich einfach an, aber was heißt es genau? Ein Beispiel:

Beispiel 1-1. Ein einleitendes Beispiel

```
<html>
    <head>
        <title>Beispiel</title>
    </head>
    <body>

        <?php
        echo "Hallo, ich bin ein PHP-Skript!";
        ?>

    </body>
</html>
```

Beachten Sie den Unterschied zu einem Skript, welches in anderen Sprachen wie Perl oder C geschrieben wurde -- anstatt ein Programm mit vielen Anweisungen zur Ausgabe von HTML zu schreiben, schreibt man einen HTML-Code mit einigen eingebetteten Anweisungen, um etwas auszuführen (z.B. um - wie oben - Text auszugeben). Der PHP-Code steht zwischen speziellen Anfangs- und Schlusstags, mit denen man in den PHP-Modus und zurück wechseln kann.

Was PHP von clientseitigen Sprachen wie Javaskript unterscheidet, ist dass der Code auf dem Server ausgeführt wird. Sollten Sie ein Skript wie das obige auf ihrem Server ausführen, würde der Besucher nur das Ergebnis empfangen, ohne die Möglichkeit herauszufinden, wie der zugrundeliegende Code aussieht. Sie können ihren Webserver auch anweisen, alle ihre HTML-Dateien mit PHP zu parsen, denn dann gibt es wirklich nichts, das dem Benutzer sagt, was sie in petto haben.

Das Beste an der Verwendung von PHP ist, dass es für Neulinge extrem simpel ist, aber auch einen riesigen Funktionsumfang für den professionellen Programmierer bietet. Scheuen Sie sich nicht, die lange Liste der PHP-Funktionen zu lesen. Sie können einsteigen, und binnen weniger Stunden bereits mit dem Schreiben von einfachen Skripten beginnen.

Auch wenn die Entwicklung von PHP auf serverseitige Skripte fokussiert ist, können Sie mit PHP weitaus mehr anstellen. Lesen Sie mehr im Abschnitt Was kann PHP?

Was kann PHP?

Alles. PHP ist hauptsächlich auf serverseitige Skripte fokussiert, weshalb Sie alles tun können, was auch ein anderes CGI Programm kann, wie z.B. Formulardaten sammeln, dynamische Inhalte für Websites generieren oder Cookies senden und empfangen. Aber PHP kann noch viel mehr.

Es gibt drei Hauptgebiete, in denen PHP Skripte genutzt werden.

- Serverseitige Skripte. Dies ist traditionelle und auch Hauptfeld von PHP. Sie benötigen dazu drei Dinge. Den PHP Parser (CGI oder Server-Modul), einen Webserver und einen Webbrowser. Sie müssen den Webserver, verbunden mit einer PHP Installation laufen lassen. Sie können Sich Ausgabe der PHP Programme über den Server mit einem Webbrowser ansehen. Für weitere Informationen lesen Sie bitte im Abschnitt Installation weiter.
- Skripte auf der Kommandozeile. Sie können auch PHP Skripte schreiben, um sie ohne einen Server oder einen Browser laufen zu lassen. Hierfür benötigen Sie nur den PHP Parser. Diese Art der Verwendung ist für die regelmäßig auszuführende Skripte mittels cron (bzw. Taskplaner unter Windows), oder einfache Aufgaben zur Verarbeitung von Text ideal. Weitere Informationen dazu finden Sie im Abschnitt Using PHP from the command line.
- Schreiben clientseitiger GUI Applikationen. PHP ist wahrscheinlich nicht die allerbeste Sprache um GUI-Applikationen zu schreiben, aber wenn Sie PHP sehr gut kennen und einige weiterführende Funktionen in Ihren clientseitigen Applikationen nutzen möchten, können Sie PHP-GTK nutzen, um derartige Programme zu schreiben. Auf diese Art haben Sie auch die Möglichkeit, Plattformübergreifende Applikationen zu schreiben. PHP-GTK ist eine Erweiterung von PHP, welche in der Hauptdistribution nicht enthalten ist. Sollten Sie daran interessiert sein, besuchen Sie die PHP-GTK Website (http://gtk.php.net/).

PHP kann auf allen gängigen Betriebssystemen verwendet werden, inkl. Linux, vielen Unix-Varianten (inkl. HP-UX, Solaris und OpenBSD), Microsoft Windows, Mac OS X, RISC OS, und wahrscheinlich anderen. PHP unterstützt auch die meisten der heute gebräuchlichen Webserver. Dies umfasst Apache, Microsoft Internet Information Server, Personal Web Server, Netscape und iPlanet Server, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd, und viele andere. Für den Großteil der Server bietet PHP ein eigenes Modul, für die anderen, welche den CGI Standard unterstützen, kann PHP als CGI Prozessor arbeiten.

So haben Sie Freiheit, PHP auf dem Betriebssystem und dem Webserver Ihrer Wahl laufen zu lassen. Weiters können Sie je nach Vorliebe prozedural, objektorientiert, oder gemischt programmieren. Auch wenn in der derzeitigen Version von PHP nicht jedes Standard OOP Feature realisiert ist, sind viele Bibliotheken und große Applikationen (inklusive der PEAR Bibliothek) exklusiv unter Verwendung von OOP Code geschrieben worden.

Mit PHP sind Sie nicht auf die Ausgabe von HTML beschränkt. Seine Fähigkeiten umfassen auch das dynamische Generieren von Bildern, PDF Dateien und Flash Animationen (mittels libswf und Ming). Sie können auch leicht jede Art von Text, wie XHTML oder irgendeine andere XML Datei ausgeben. PHP kann diese Dateien automatisch generieren und im Dateisystem speichern, anstatt diese nur auszugeben bzw. auszudrucken, und formt auch serverseitigen Zwischenspeicher Ihrer dynamischen Inhalte.

Vielleicht die größte und bemerkenswerteste Stärke von PHP ist seine Unterstützung für eine breite Masse von Datenbanken. Eine datenbankgestützte Website zu erstellen ist unglaublich einfach. Die folgenden Datenbanken werden zur Zeit unterstützt:

Adabas D	Ingres	Oracle (OCI7 und OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (nur Lesezugriff)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Wir haben auch die Erweiterung DBX zur Datenbankabstatraktion, welche Ihnen die transparente Verwendung irgendeiner von dieser Erweiterung unterstützten Datenbank erlaubt. Weiters

unterstützt PHP ODBC, den Open Database Connection Standard, mit welchem Sie sich zu jeder anderen, diesen Weltstandard unterstützenden Datenbank verbinden können.

PHP unterstützt auch die Kommunikation mit anderen Services, welche Protokolle wie LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (auf Windows) und unzählige andere unterstützen. Sie können auch einfache Netzwerk Sockets öffnen, und unter Verwendung irgendeines Protokolls kommunizieren. PHP unterstützt auch WDDX (Web Distributed Data Exchange) zum Austausch komplexer Daten zwischen Programmiersprachen. Da wir gerade vom Zusammenwirken sprechen: PHP bietet auch Unterstützung zur Instantiierung von Java Objekten, und deren transparente Verwendung als PHP Objekte. Sie können auch unsere CORBA Erweiterung verwenden, um auf entfernte Objekte zuzugreifen.

PHP verfügt über äußerst hilfreiche Textverarbeitungsfunktionen, von den regulären Ausdrücken (POSIX erweitert oder Perl), bis zum Parsen von XML Dokumenten. Für den Zugriff und das Parsen von XML Dokumenten unterstützen wir die Standards SAX und DOM. Sie können unsere XSLT Erweiterung verwenden, um XML Dokumente zu transformieren.

Wenn Sie PHP im Bereich des E-Commerce nutzen, werden Sie Funktionen für Cybercash Payment, CyberMUT, VeriSign Payflow Pro und CCVS für Ihre Onlineprogramme zur Zahlungsabwicklung zu schätzen lernen.

Wir haben viele andere interessante Erweiterungen, wie mnoGoSearch für Suchmaschinen, die IRC Gateway Funktionen, viele Komprimierungswerkzeuge (gzip, bz2), Kalenderumrechnung, Übersetzung...

Wie Sie sehen können reicht diese Seite nicht aus, um alle Möglichkeiten und Vorteile von PHP aufzulisten. Lesen Sie im Abschnitt Installation weiter, und konsultieren Sie auch die Funktionsreferenz für weitere Erläuterungen der einzelnen hier erwähnten Erweiterungen.

Kapitel 2. Installation

Download der aktuellsten Version

Den Quellcode sowie die Binärdistributionen für die verschiedenen Plattformen (inklusive Windows) erhalten Sie unter `http://www.php.net/`. Wir empfehlen, den Download der Distributionen von dem Ihnen am nächsten liegenden Mirror (http://www.php.net/mirrors.php) durchzuführen.

Installation auf UNIX Systemen

Dieser Abschnitt wird Sie durch die Konfiguration und Installation von PHP auf UNIX Systemen führen. Bevor Sie mit dem Prozess beginnen, lesen Sie bitte alle für Ihre Plattform oder Webserver relevanten Sektionen.

Erforderliche Kenntnisse und Software:

- Grundlegende UNIX-Kenntnisse (Sie sollten in der Lage sein, mit "make" und einem C-Compiler umzugehen, wenn Sie kompilieren)
- Einen ANSI C Compiler (wenn Sie kompilieren)
- flex (zum kompilieren)
- bison (zum kompilieren)
- Einen Web Server
- Alle modulspezifischen Komponenten (wie z.B. gd, pdf libs, etc.)

Es gibt verschiedene Wege, PHP auf der Unix Plattform zu installieren. Entweder mit einem Kompilier- und Configure- Prozess, oder über verschiedene Methoden mittels vorgefertigter Pakete. Diese Dokumentation ist hauptsächlich auf den Prozess des Kompilierens und Konfigurierens von PHP fokussiert.

Der Anfang des PHP Installations- und Konfigurationsprozesses wird von den Eingabeoptionen für das `configure` Skript gesteuert. Diese Seite umreißt die Verwendung der meisten allgemeinen Optionen, aber es gibt noch viele andere als Spielraum. Eine umfassende Übersicht finden Sie in der Kompletten Liste der Konfigurationsoptionen. Es gibt verschiedene Wege, PHP zu installieren:

- Als ein Apache Modul
- Als ein fhttpd Modul
- Zur Verwendung mit AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, oder Zeus.
- Als ein CGI Programm

Apache Modul - Kurzanleitung

PHP kann auf einigen verschiedenen Wegen Kompiliert werden, aber eine der populärsten ist die als Apache Modul. Es folgt ein kurzer Überblick über die Installation.

Beispiel 2-1. Kurzanleitung zur Installation von PHP 4 (Apache Modul Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install
15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
17. Editieren Sie nun Ihre httpd.conf oder srm.conf, und fügen Sie hinzu:
      AddType application/x-httpd-php .php

18. Starten Sie den Apache Web Server neu. (Sie müssen den Server beenden
    und neu starten, ein einfacher Reload mittels HUP oder USR1 Signal
    reicht nicht aus.)
```

Building

Wenn PHP konfiguriert ist, können Sie die ausführbare CGI Datei oder PHP-Bibliothek erstellen. Das Kommando **make** sollte das für Sie erledigen. Sollte dies scheitern und Sie nicht erkennen können warum, lesen Sie in der Problem- Sektion weiter.

Unix/Linux Installation

Dieser Teil enthält Anmerkungen und Hinweise zur Installation von PHP auf Linux Distributionen

Pakete verwenden

Viele Linux Distributionen haben eine Art Paketinstallation, wie z.B. RPM. Das kann beim Setup einer Standardkonfiguration helfen, aber wenn Sie andere Features (wie einen Secure Server, oder einen anderen Datenbanktreiber) benötigen könnte es sein, dass ein Build von PHP und/oder Ihres Webservers nötig ist. Wenn Sie mit dem Bilden und Kompilieren Ihrer Software nicht vertraut sind, sollten Sie sich nach einer von jemand anders bereits erstellten Paketversion von PHP mit den von Ihnen benötigten Features umsehen.

Unix/HP-UX Installationen

Diese Sektion enthält Anmerkungen und Hinweise zur Installation von PHP auf HP-UX Systemen.

Beispiel 2-2. Installationsanleitung für HP-UX 10

```
Von: paul_mckay@clearwater-it.co.uk
04-Jan-2001 09:49
(Diese Tipps sind für PHP 4.0.4 und Apache v1.3.9)

Sie wollen also PHP und Apache auf einer HP-UX 10.20 installieren?

1. Sie brauchen gzip, holen Sie sich eine Binärdistribution von
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z
dekomprimieren Sie die Datei, und installieren Sie sie mit swinstall

2. Sie brauchen gcc, holen Sie sich eine Binärdistribution von
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz
gunzip, und installieren Sie sie mit swinstall.

3. Sie brauchen die GNU binutils, holen Sie sich eine Binärdistribution von
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz
gunzip, und installieren Sie sie mit swinstall.

4. Nun benötigen Sie bison, holen Sie sich eine Binärdistribution von
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz
installieren Sie sie wie oben

4. Nun benötigen Sie flex, holen Sie sich den Source von einem der
http://www.gnu.org mirrors. Es ist in dem non-gnu
Verzeichnis der FTP-Site.
Downloaden Sie die Datei, gunzip, dann tar -xvf ’en Sie sie. Gehen Sie in das erstellte
Verzeichnis flex, und führen ein ./configure aus, dann make, und dann make install.

Sollten dabei Fehler auftreten liegt es vielleicht daran, dass gcc etc.
nicht im PATH sind. Ist das der Fall, fügen Sie diese bitte dem PATH hinzu.

OK, nun zum schwierigen Teil:

6. Laden Sie die Sources von PHP und Apache herunter.

7. Führen Sie gunzip und tar -xvf an ihnen aus.

Wir müssen einige Dateien ändern, dass sie richtig kompiliert werden können.

8. Erst muss die Konfigurationsdatei geändert werden, denn es scheint als
würde sie aus den Augen verlieren, dass sie auf einer hpux Maschine ist. Es
gibt sicher einen besseren Weg dies zu tun, aber einfacher Weg ist es,
    lt_target=hpux10.20
in die Zeile 47286 des configure Skriptes zu schreiben.

9. Als nächstes müssen wir die Apache GuessOS Datei ändern. Unter
apache_1.3.9/src/helpers ändern Sie die Zeile 89 von
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
auf:
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"
```

```
10. Sie können PHP nicht als Shared Object unter HP-UX installieren, weshalb Sie es
statisch kompilieren müssen; folgen Sie einfach den Anweisungen auf der Apache Seite.

11. Das Kompilieren PHP und Apache sollte OK gehen, aber Apache wird nicht starten.
Sie müssen einen neuen User für Apache einrichten, z.B. www oder apache. Dann ändern
Sie die Zeilen 252 und 253 der conf/httpd.conf in Apache so, dass statt
    User nobody
    Group nogroup
etwas ähnliches steht wie:
    User www
    Group sys

Das hat den Grund, dass Sie Apache unter HP-UX nicht als nobody laufen
lassen können. Jetzt sollten Apache und PHP laufen.

Ich hoffe das hilft jemandem,
Paul Mckay.
```

Unix/Solaris Installation

Diese Sektion enthält Anmerkungen und Hinweise zur Installation von PHP auf Solaris Systemen.

Benötigte Software

Solaris Installationen mangelt es oft an C Compilern und deren zugehörigen Tools. Sie benötigen folgende Software:

- gcc (empfohlen, andere C Compiler können ebenfalls funktionieren)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar

Zusätzlich werden Sie für die Installation (und vielleicht auch zum Kompilieren) eine andere, für Ihre Konfiguration spezifische Software benötigen, wie z.B. Oracle oder MySQL.

Pakete verwenden

Sie können den Installationsprozess auf Solaris vereinfachen, indem Sie zur Installation der meisten benötigten Komponenten pkgadd verwenden.

Unix/OpenBSD Installation

Diese Sektion enthält Anmerkungen und Hinweise zur Installation von PHP auf OpenBSD (http://www.openbsd.org/).

Ports verwenden

Dies ist die empfohlene Methode PHP auf OpenBSD zu installieren, da sie die aktuellen Patches und Security Fixes enthält. Um diese Methode zu verwenden, benötigen Sie einen recent ports tree (http://www.openbsd.org/ports.html). Dann bestimmen Sie einfach die zu installierenden Optionen, und führen das Kommando **make install** aus. Das folgende Beispiel zeigt wie es gemacht wird.

Beispiel 2-3. OpenBSD Ports Installationsbeispiel

```
$ cd /usr/ports/www/php4
$ make show VARNAME=FLAVORS
 (wählen Sie gewünschten Installationsoptionen aus der liste)
$ FLAVOR="imap gettext ldap mysql gd" make install
$ php4-enable
```

Pakete verwenden

Für Ihre Version von OpenBSD (http://www.openbsd.org/) sind vorkompilierte Pakete verfügbar. Diese integrieren sich automatisch in die mit dem OS installierten Apache Version. Da es jedoch eine große Anzahl Optionen (genannt *flavors*) für dieses Port gibt, ziehen Sie es vielleicht vor, den Source unter Verwendung des Ports Tree zu kompilieren. Weitere Informationen über die verfügbaren Pakete finden Sie in der Manual Page packages(7) (http://www.openbsd.org/cgi-bin/man.cgi?query=packages)

Unix/Mac OS X Installationen

Diese Sektion enthält Anmerkungen und Hinweise zur Installation von PHP auf Mac OS X Server.

Pakete verwenden

Es gibt ein paar vorgefertigte und kompilierte Pakete von PHP für Mac OS X. Das ist hilfreich, um eine Standardkonfiguration aufzusetzen, doch sollten Sie eine davon abweichende Konfiguration benötigen (wie z.B. einen Secure Server oder einen anderen Datenbanktreiber), müssen Sie evtl. PHP und/oder Ihren Webserver selbst bauen. Wenn Sie mit dem Bilden und Kompilieren Ihrer

Software nicht vertraut sind, sollten Sie sich nach einer von jemand anders bereits erstellten Paketversion von PHP mit den von Ihnen benötigten Features umsehen.

Kompilieren für OS X Server

Es gibt zwei leicht unterschiedliche Versionen von Mac OS X, nämlich einen Client und einen Server. Die folgende Beschreibung ist für OS X Server.

Beispiel 2-4. Mac OS X Server Installation

```
1. Beschaffen Sie sich die aktuellen Distributionen von Apache und PHP
2. Untar'en Sie sie, und starten configure an Apache wie folgt
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache

4. Sie können auch folgende Zeile hinzufügen:
    setenv OPTIM=-O2
    Wenn Sie wollen, dass der Compiler Optimierungen durchführt.

5. Wechseln Sie in das Verzeichnis mit dem PHP Source und führen configure aus
    ./configure --prefix=/usr \
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12

    Wenn Sie andere Ergänzungen haben (MySQL, GD, etc.), fügen Sie diese bitte
    hier hinzu. Geben Sie bei der Option --with-apache den Pfad zu dem
    Verzeichnis mit Ihrem Apache Source an, z.B. "/src/apache_1.3.12".
6. make
7. make install
    Dies fügt Ihrem Apache Source Verzeichnis das Verzeichnis
    src/modules/php4 hinzu.

8. Nun konfigurieren Sie Apache neu, incl. PHP 4 Modul.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache \
    --activate-module=src/modules/php4/libphp4.a

    Sollten Sie eine Meldung erhalten, dass libmodphp4.a nicht mehr gültig ist,
```

```
    wechseln Sie in Ihrem Apache Source Verzeichnis zu src/modules/php4, und
    führen folgenden Befehl aus:

    ranlib libmodphp4.a

    Dann gehen Sie zurück in das Hauptverzeichnis des Apache Source, und
    führen obigen configure Befehl erneut aus. Das aktualisiert die Linktabelle.

9. make

10. make install

11. Kopieren Sie die Datei php.ini-dist von Ihrem Verzeichnis mit dem PHP 4
    Source in Ihr "bin" Verzeichnis, und benennen Sie sie um:
    cp php.ini-dist /usr/local/bin/php.ini

    oder (wenn Sie kein lokales Verzeichnis haben)

    cp php.ini-dist /usr/bin/php.ini
```

Andere Beispiele für Mac OS X Client (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) und Mac OS X Server (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) finden Sie bei Stepwise (http://www.stepwise.com/).

Kompilieren für MacOS X Client

Diese Tipps wurden freundlicherweise von Marc Liyanage (http://www.entropy.ch/software/macosx) zur Verfügung gestellt.

Das PHP Modul für den Apache Webserver ist in Mac OS X inkludiert. Diese Version unterstützt die MySQL und PostgreSQL Datenbanken.

Anmerkung: Vorsicht wenn Sie dies tun, Sie könnten Ihren Apache Webserver vermurksen!

Installationsschritte:

- 1. Öffnen Sie ein Terminalfenster
- 2. Geben Sie "wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz" ein, und warten Sie bis der Download beendet ist
- 3. Geben Sie "gunzip libphp4.so.gz" ein
- 4. Geben Sie "sudo apxs -i -a -n php4 libphp4.so" ein

Nun geben Sie "`sudo open -a TextEdit /etc/httpd/httpd.conf`" ein. TextEdit öffnet daraufhin die Konfigurationsdatei des Webservers. Suchen Sie diese beiden Zeilen in Richtung Dateiende: (Benutzen Sie den Befehl Find)

```
*    #AddType application/x-httpd-php .php
*    #AddType application/x-httpd-php-source .phps
```

Entfernen Sie die beiden Rautezeichen (#), speichern die Datei, und beenden TextEdit.

Starten Sie mit "`sudo apachectl graceful`" den Webserver neu.

PHP sollte nun laufen. Sie können dies testen, indem Sie eine Datei in Ihrem Verzeichnis "Sites" erstellen, z.B. "test.php", folgende Zeile hineinschreiben: "`<?php phpinfo() ?>`".

Nun öffnen Sie `127.0.0.1/~your_username/test.php` in Ihrem Webbrowser. Sie sollten nun eine Tabelle mit Informationen über das PHP Modul sehen.

Komplette Liste der Konfigurationsoptionen

Anmerkung: Diese Optionen werden nur zur Kompilierzeit genutzt. Wenn Sie die Laufzeitkonfiguration von PHP ändern möchten, lesen Sie bitte unter Configuration nach.

Nachstehend finden Sie die komplette Liste von Optionen, welche von den PHP 3 und PHP 4 `configure` Skripten unterstützt, und zum Kompilieren in einer Unix-ähnlichen Umgebung verwendet werden. Die Anmerkungen zeigen, ob eine Option in PHP 3, PHP 4, oder in beiden Versionen verfügbar ist. Es gibt auch einige Optionen, deren Namen in PHP 3 und PHP 4 differieren, aber den gleichen Effekt haben. Diese Einträge haben Querverweise zueinander, sodass Sie bei Problemen mit Optionen von PHP 3 schnell erkennen können, ob sich deren Namen geändert haben.

- Datenbank
- Ecommerce
- Grafik
- Verschiedenes
- Networking
- PHP Eigenschaften
- Server
- Text und Sprache
- XML

Datenbank

`--with-adabas[=DIR]`

PHP 3, PHP 4: Inkludiert Adabas D Unterstützung. DIR ist das Adabas Installationsverzeichnis. Default ist /usr/local.

Adabas Homepage (http://www.software-ag.com/adabasd/)

`--enable-dba=shared`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: DBA als shared module erstellen

`--enable-dbx`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert DBX Unterstützung

`--enable-dbase`

PHP 3: Ist in PHP 3 nicht verfügbar; verwenden Sie stattdessen --with-dbase

PHP 4: Inkludieren der bundled dbase Bibliothek. Es sind keine externen Bibliotheken erforderlich.

`--with-dbase`

PHP 3: Inkludieren der bundled dbase Bibliothek. Es sind keine externen Bibliotheken erforderlich.

PHP 4: Ist nicht verfügbar; verwenden Sie stattdessen --enable-dbase.

`--with-db2[=DIR]`

PHP 3, PHP 4: Inkludiert Berkeley DB2 Unterstützung

`--with-db3[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Berkeley DB3 Unterstützung

`--with-dbm[=DIR]`

PHP 3, PHP 4: Inkludiert DBM Unterstützung

`--with-dbmaker[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert DBMaker Unterstützung. DIR ist das DBMaker base install directory, Default ist das Verzeichnis, in dem die letzte Version von DBMaker installiert ist (wie z.B. /home/dbmaker/3.6).

`--with-empress[=DIR]`

PHP 3, PHP 4: Inkludiert Empress Unterstützung. DIR ist das Empress base install directory, Default ist $EMPRESSPATH

`--enable-filepro`

PHP 3: Ist nicht verfügbar; verwenden Sie stattdessen --with-filepro.

PHP 4: Aktiviert die bundled read-only filePro Unterstützung. Es sind keine externen Bibliotheken erforderlich.

`--with-filepro`

PHP 3: Aktiviert die bundled read-only filePro Unterstützung. Es sind keine externen Bibliotheken erforderlich.

PHP 4: Ist nicht verfügbar; verwenden Sie stattdessen --enable-filepro.

`--with-fbsql[=DIR]`

PHP 3: Ist nicht verfügbar.

PHP 4: Inkludiert FrontBase SQL Unterstützung. DIR ist das FrontBase base install directory, Default ist das übliche Frontbase Installationsverzeichnis. Das übliche Installationsverzeichnis ist von Ihrem OS abhängig: Solaris: `/opt/FrontBase`, WinNT: `\usr\FrontBase`, Linux: `/usr/frontbase`, Mac OSX: `/Library/FrontBase`.

`--with-gdbm[=DIR]`

PHP 3, PHP 4: Inkludiert GDBM Unterstützung

`--with-hyperwave`

PHP 3, PHP 4: Inkludiert Hyperwave Unterstützung

`--with-ibm-db2[=DIR]`

PHP 3, PHP 4: Inkludiert IBM DB2 Unterstützung. DIR ist das DB2 base install directory, default ist `/home/db2inst1/sqllib`.

IBM DB2 home page (http://www.ibm.com/db2/)

`--with-informix[=DIR]`

PHP 3, PHP 4: Inkludiert Informix Unterstützung. DIR ist das Informix base install directory, Hat keinen Defaultwert.

`--with-ingres[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Ingres II Unterstützung. DIR ist das Ingres base directory (default ist /II/ingres)

`--with-interbase[=DIR]`

PHP 3, PHP 4: Inkludiert InterBase Unterstützung. DIR ist das InterBase base install directory, default ist `/usr/interbase`.

Interbase Funktionen

Interbase Homepage (http://www.interbase.com/)

`--with-ldap[=DIR]`

PHP 3: Inkludiert LDAP Unterstützung. DIR ist das LDAP base install directory. Default ist `/usr` und `/usr/local`

PHP 4: Inkludiert LDAP Unterstützung. DIR ist das LDAP base install directory.

Dies bietet LDAP (Lightweight Directory Access Protocol Unterstützung). Der Parameter ist das LDAP base install directory, default ist `/usr/local/ldap`.

Mehr Informationen über LDAP finden Sie in RFC1777 (http://www.faqs.org/rfcs/rfc1777.html) und RFC1778 (http://www.faqs.org/rfcs/rfc1778.html).

`--with-msql[=DIR]`

PHP 3, PHP 4: Inkludiert mSQL Unterstützung. Der Parameter zu dieser Option ist das mSQL install directory, default ist `/usr/local/Hughes`. Das ist das Default Verzeichnis der mSQL 2.0 Distribution. **configure** erkennt automatisch welche Version von mSQL Sie benutzen. PHP unterstützt sowohl 1.0 als auch 2.0, wenn Sie jedoch PHP mit mSQL 1.0 kompilieren, können Sie nur mSQL 1.0 Datenbanken benutzen, und umgekehrt.

Siehe auch mSQL Konfiguration Direktiven in der Konfigurationsdatei.

mSQL Homepage (http://www.hughes.com.au/)

`--with-mysql[=DIR]`

PHP 3: Inkludiert MySQL Unterstützung. DIR ist das MySQL base install directory, als default werden einige für MySQL Dateien übliche Stellen durchsucht.

PHP 4: Inkludiert MySQL Unterstützung. DIR ist das MySQL base directory. Wenn nicht spezifiziert, wird die bundled MySQL Bibliothek verwendet. Diese Option ist standardmäßig aktiviert.

Siehe auch MySQL Konfiguration Direktiven in der Konfigurationsdatei.

MySQL Homepage (http://www.mysql.com/)

`--with-ndbm[=DIR]`

PHP 3, PHP 4: Inkludiert NDBM Unterstützung.

`--with-ovrimos`

PHP 3, PHP 4: Inkludiert Ovrimos Unterstützung.

`--with-oci8[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Oracle-oci8 Unterstützung. Default DIR ist ORACLE_HOME.

`--with-oracle[=DIR]`

PHP 3: Inkludiert Oracle Datenbankunterstützung. DIR ist Oracle's Home directory, default ist $ORACLE_HOME.

PHP 4: Inkludiert Oracle-oci7 Unterstützung. Default DIR ist ORACLE_HOME.

Inkludiert Oracle Unterstützung, wurde getestet und sollte zumindest mit den Oracle Versionen 7.0 bis 7.3 funktionieren. Der Parameter ist das ORACLE_HOME Verzeichnis. Sie müssen diesen Parameter nicht angeben, wenn Ihre Oracle Umgebung aufgesetzt ist.

Oracle home page (http://www.oracle.com/)

--with-pgsql[=DIR]

PHP 3: Inkludiert PostgresSQL Unterstützung. DIR ist das PostgresSQL base install directory, default ist `/usr/local/pgsql`.

PHP 4: Inkludiert PostgreSQL Unterstützung. DIR ist das PostgreSQL base install directory, default ist `/usr/local/pgsql`. Setzen Sie DIR auf shared um als ein dl zu builden, oder shared,DIR um als ein dl zu builden, und DIR zu spezifizieren.

Siehe auch Postgres Konfiguration Direktiven in der Konfigurationsdatei.

PostgreSQL home page (http://www.postgresql.org/)

--with-solid[=DIR]

PHP 3, PHP 4: Inkludiert Solid Unterstützung. DIR ist das Solid base install directory, default ist /usr/local/solid

Solid Homepage (http://www.solidtech.com/)

--with-sybase-ct[=DIR]

PHP 3, PHP 4: Inkludiert Sybase-CT Unterstützung. DIR ist das Sybase home directory, default ist /home/sybase.

Siehe auch Sybase-CT Konfigurations Direktiven in der Konfigurationsdatei.

--with-sybase[=DIR]

PHP 3, PHP 4: Inkludiert Sybase-DB Unterstützung. DIR ist das Sybase home directory, default ist `/home/sybase`.

Siehe auch Sybase Konfigurations Direktiven in der Konfigurationsdatei.

Sybase Homepage (http://www.sybase.com/)

--with-openlink[=DIR]

PHP 3, PHP 4: Inkludiert OpenLink ODBC Unterstützung. DIR ist das OpenLink base install directory, default ist /usr/local/openlink. Seit PHP 4.0.6 ist diese Option nicht mehr gültig. Verwenden Sie die Option *--with-iodbc*, wenn Sie das ODBC System von OpenLink Software benutzen wollen.

OpenLink Software's Homepage (http://www.openlinksw.com/)

--with-iodbc[=DIR]

PHP 3, PHP 4: Inkludiert iODBC Unterstützung. DIR ist das iODBC base install directory, default ist `/usr/local`.

Dieses Feature wurde erst für iODBC Driver Manager entwickelt, einem frei verfügbaren ODBC Treibermanager, welcher unter vielen UNIX-Derivaten läuft.

FreeODBC Homepage (http://users.ids.net/~bjepson/freeODBC/) or iODBC Homepage (http://www.iodbc.org/)

--with-custom-odbc[=DIR]

PHP 3, PHP 4: Inkludiert Unterstützung für eine spezielle ODBC Bibliothek. Der Parameter ist das Basisverzeichnis, default ist `/usr/local`.

Diese Option setzt voraus, dass Sie CUSTOM_ODBC_LIBS definiert haben, wenn Sie das configure Skript starten. Sie brauchen auch einen gültigen odbc.h header in Ihrem Include Pfad. Wenn Sie keinen haben, erstellen Sie ihn und inkludieren Sie Ihren spezifischen Header von dort. Ihr Header könnte auch ein paar zusätzliche Definitionen benötigen, speziell wenn es sich um eine Multiplattform handelt. Definieren Sie diese in CFLAGS.

Zum Beispiel können Sie Sybase SQL Anywhere auf QNX wie folgt verwenden:

```
CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc"
./configure --with-custom-odbc=/usr/lib/sqlany50
```

--disable-unified-odbc

PHP 3: Unified ODBC Unterstützung verhindern. Ist nur angebracht, wenn iODBC, Adabas, Solid, Velocis oder eine spezielle ODBC Schnittstelle aktiviert ist.

PHP 4: Ist unter PHP 4 nicht verfügbar.

Das Unified ODBC Modul ist eine einheitliche Schnittstelle zu allen Datenbanken mit ODBC-basierten Schnittellen, wie z.B. Solid, IBM DB2 und Adabas D. Es arbeitet auch mit normalen ODBC Bibliotheken. Es wurde getestet mit iODBC, Solid, Adabas D, IBM DB2 und Sybase SQL Anywhere getestet. Es benötigt eine (und nur eine) dieser Erweiterungen oder die Velocis Erweiterung aktiviert, oder eine spezielle ODBC Bibliothek spezifiziert. Diese Option nur angebracht, wenn eine der folgenden Optionen verwendet wird: --with-iodbc, --with-solid, --with-ibm-db2, --with-adabas, --with-velocis, or --with-custom-odbc.

Siehe auch Unified ODBC Konfiguration Direktiven in der Konfigurationsdatei.

--with-unixODBC[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert unixODBC Unterstützung. DIR ist das unixODBC base install directory, default ist /usr/local.

--with-velocis[=DIR]

PHP 3, PHP 4: Inkludiert Velocis Unterstützung. DIR ist das Velocis base install directory, default ist /usr/local/velocis.

Velocis Homepage (http://www.birdstep.com/database_technology/rdm_server.php3)

Ecommerce

--with-ccvs[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Kompiliert CCVS Unterstützung in PHP 4. Bitte geben Sie Ihr CCVS base install directory als DIR an.

--with-cybermut[=DIR]

PHP 3: Diese Option ist für PHP 3 nicht verfügbar.

PHP 4: Cybermut-Unterstützung in PHP 4 einkompilieren. DIR ist das Cybermut SDK Basisverzeichnis, das mindestens die beiden Dateien `libcm-mac.a` und `cm-mac.h` enthält.

--with-mck[=DIR]

PHP 3: Inkludiert Cybercash MCK Unterstützung. DIR ist das cybercash mck build directory, default ist `/usr/src/mck-3.2.0.3-linux`. Hilfe finden Sie in `extra/cyberlib`.

PHP 4: Ist nicht verfügbar; verwenden Sie stattdessen --with-cybercash .

--with-cybercash[=DIR]

PHP 3: Ist nicht verfügbar; verwenden Sie stattdessen --with-mck .

PHP 4: Inkludiert CyberCash Unterstützung. DIR ist das CyberCash MCK install directory.

--with-pfpro[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Verisign Payflow Pro Unterstützung

Graphics

--enable-freetype-4bit-antialias-hack

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Unterstützung für FreeType2 (experimentell).

--with-gd[=DIR]

PHP 3: Inkludiert GD Unterstützung (DIR ist das GD install dir).

PHP 4: Inkludiert GD Unterstützung (DIR ist das GD install dir). Setzen Sie DIR auf shared um als ein dl zu builden, oder shared,DIR um als ein dl zu builden und DIR trotzdem zu spezifizieren.

`--without-gd`

PHP 3, PHP 4: Deaktiviert GD Unterstützung.

`--with-imagick[=DIR]`

PHP 3: Inkludiert ImageMagick Unterstützung. DIR ist das Installationsverzeichnis, und wenn Sie es nicht angeben, versucht es PHP selbst zu finden. [experimentell]

PHP 4: Ist in PHP 4 nicht verfügbar

`--with-jpeg-dir[=DIR]`

PHP 3: jpeg dir für pdflib 2.0

PHP 4: jpeg dir für pdflib 3.x and 4.x

`--with-png-dir[=DIR]`

PHP 3: OIst in PHP 3 nicht verfügbar

PHP 4: png dir für pdflib 3.x und 4.x

`--enable-t1lib`

PHP 3: Inkludiert t1lib Unterstützung.

PHP 4: Ist nicht verfügbar; verwenden Sie stattdessen --with-t1lib.

`--with-t1lib[=DIR]`

PHP 3: Ist nicht verfügbar; verwenden Sie stattdessen --enable-t1lib.

PHP 4: Inkludiert T1lib Unterstützung.

`--with-tiff-dir[=DIR]`

PHP 3: tiff dir für pdflib 2.0

PHP 4: tiff dir für pdflib 3.x und 4.x

`--with-ttf[=DIR]`

PHP 3, PHP 4: Inkludiert FreeType Unterstützung

`--with-xpm-dir[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: xpm dir für gd-1.8+

Verschiedenes

Diese Optionen werden mit der Zeit - wo angebracht - klassifiziert.

--with-gmp

PHP 3, PHP 4 : Inkludiert GMP Unterstützung.

--disable-bcmath

PHP 3: Kompiliert ohne BC arbitrary precision math Funktionen. Diese Funktionen erlauben Ihnen mit Zahlen außerhalb des bei regulären Integern und Floats erlaubten Bereiches zu arbeiten. Siehe auch BCMath Arbitrary Precision Mathematics Functions für weitere Informationen.

PHP 4: Ist nicht verfügbar; bcmath ist nicht standardmäßig einkompiliert. Verwenden Sie --enable-bcmath, um es zu integrieren.

--disable-display-source

PHP 3: Kompiliert ohne Unterstützung für Source-Anzeige

PHP 4: Ist in PHP 4 nicht verfügbar

--disable-libtool-lock

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Vermeiden Sie locking (könnte parallele Builds zerstören)

--disable-pear

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Installiert kein PEAR

--disable-pic

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Verhindert PIC für shared objects

--disable-posix

PHP 3: Ist nicht verfügbar; verwenden Sie stattdessen --without-posix.

PHP 4: Verhindert POSIX-ähnliche Funktionen

--enable-pcntl

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert Prozeßsteuerungsfunktionen. (fork, waitpid, signal, etc.)

`--disable-rpath`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Verhindert die Übergabe weiterer runtime library search paths

`--disable-session`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Verhindert Session Unterstützung

`--enable-bcmath`

PHP 3: Ist in PHP 3 nicht verfügbar; bcmath ist standardmäßig einkompiliert. Verwenden Sie --disable-bcmath, um dies zu verhindern.

PHP 4: Kompiliert mit bc style precision math Funktionen. Siehe README-BCMATH für Instruktionen, wie dieses installiert wird. Diese Funktionen erlauben Ihnen, mit Zahlen außerhalb des bei regulären Integern und Floats erlaubten Bereiches zu arbeiten. Weitere Informationen finden Sie in BCMath Arbitrary Precision Mathematics Functions.

`--enable-c9x-inline`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert C9x-inline semantics

`--enable-calendar`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Unterstützung für Kalenderumrechnung

`--enable-debug`

PHP 3, PHP 4: Kompiliere mit debugging symbols.

`--enable-debugger`

PHP 3: Kompiliere mit remote debugging functions

PHP 4: Ist in PHP 4 nicht verfügbar

`--enable-discard-path`

PHP 3, PHP 4: Ist diese Option aktiviert, kann das CGI Binary sicher außerhalb des Web Verzeichnisbaumes platziert werden, und die Leute können die .htaccess Security nicht umgehen.

`--enable-dmalloc`

PHP 3, PHP 4: Aktiviert dmalloc

`--enable-exif`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert exif Unterstützung

`--enable-experimental-zts`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Diese Option wird wahrscheinlich Ihren Build zerstören

`--enable-fast-install[=PKGS]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Optimiert für schnelle Installation [default=yes]

`--enable-force-cgi-redirect`

PHP 3, PHP 4: Aktiviert die Sicherheitsprüfung für interne Server Redirects. Sie sollten dies verwenden, wenn Sie die CGI Version mit Apache einsetzen.

`--enable-inline-optimization`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Wenn Sie viel Speicher haben und gcc verwenden, können Sie dies probieren.

`--enable-libgcc`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Linkt explizit gegen libgcc

`--enable-maintainer-mode`

PHP 3, PHP 4: Aktiviert make Regeln und Abhängigkeiten, welche für den unbeschwerten Installierer nicht hilfreich (und manchmal verwirrend) sind.

`--enable-mbstr-enc-trans`

PHP 4: Aktiviert http input character automatic detection and translation für multi-byte character encodings.

Warnung

Diese Option ist nur für PHP 4.0.6 oder höher verfügbar.

`--enable-mbstring`

PHP 4: Aktiviert mit multi-byte character encoding verwandte Funktionen.

Warnung

Diese Option ist nur für PHP 4.0.6 oder höher verfügbar.

`--enable-memory-limit`

PHP 3, PHP 4: Kompiliert mit Memory Limit Unterstützung. [default=no]

`--enable-safe-mode`

PHP 3, PHP 4: Aktiviert Safe Mode standardmäßig.

`--enable-satellite`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert CORBA Unterstützung über Satellite (Benötigt ORBit)

`--enable-shared[=PKGS]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Build mit shared libraries [default=yes]

`--enable-sigchild`

PHP 3, PHP 4: Aktiviert PHP's eigenen SIGCHLD Handler.

`--enable-static[=PKGS]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Build mit static libraries [default=yes]

`--enable-sysvsem`

PHP 3, PHP 4: Aktiviert die Unterstützung für System V semaphore Unterstützung.

`--enable-sysvshm`

PHP 3, PHP 4: Aktiviert die Unterstützung für System V shared memory support

`--enable-trans-sid`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert Transparent Session ID Propagation

`--with-cdb[=DIR]`

PHP 3, PHP 4: Inkludiert CDB Unterstützung

--with-config-file-path=PATH

PHP 3: Setzt den Pfad, in welchem nach der php3.ini gesucht werden soll. Default ist `/usr/local/lib`.

PHP 4: Setzt den Pfad, in welchem nach der `php.ini` gesucht werden soll. Default ist `/usr/local/lib`

--with-cpdflib[=DIR]

PHP 3: Inkludiert ClibPDF Unterstützung. DIR ist das ClibPDF install directory, default ist /usr/local.

PHP 4: Inkludiert cpdflib Unterstützung (benötigt cpdflib >= 2). DIR ist das cpdfllib Installatiionsverzeichnis, default ist /usr.

--with-esoob[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Easysoft OOB Unterstützung. DIR ist das OOB base install directory, default ist /usr/local/easysoft/oob/client.

--with-exec-dir[=DIR]

PHP 3, PHP 4: Erlaube Executables in DIR nur, wenn im Safe Mode. Default ist /usr/local/php/bin

--with-fdftk[=DIR]

PHP 3, PHP 4: Inkludiert fdftk Unterstützung. DIR ist das fdftk install directory, default ist /usr/local.

--with-gnu-ld

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Nimmt an, dass der C Compiler GNU ld verwendet [default=no]

--with-icap[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert ICAP Unterstützung

--with-imap[=DIR]

PHP 3, PHP 4: Inkludiert IMAP Unterstützung. DIR ist das IMAP include, und das c-client.a directory.

--with-imsp[=DIR]

PHP 3: Inkludiert IMSP Unterstützung (DIR ist das IMSP include dir und libimsp.a dir).

PHP 4: Ist in PHP 4 nicht verfügbar

`--with-java[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Java Unterstützung. DIR ist das base install directory für das JDK. Diese Extension kann nur als eine shared dl erstellt werden.

`--with-kerberos[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Kerberos Unterstützung in IMAP.

`--with-mcal[=DIR]`

PHP 3, PHP 4: Inkludiert MCAL Unterstützung.

`--with-mcrypt[=DIR]`

PHP 3, PHP 4: Inkludiert mcrypt Unterstützung. DIR ist das mcrypt install directory.

`--with-mhash[=DIR]`

PHP 3, PHP 4: Inkludiert mhash Unterstützung. DIR ist das mhash install directory.

`--with-mm[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert mm Unterstützung für Session Storage

`--with-mod_charset`

PHP 3, PHP 4: Aktiviert Transfer Tables für mod_charset (Rus Apache).

`--with-pdflib[=DIR]`

PHP 3: Inkludiert pdflib Unterstützung (getestet mit 0.6 und 2.0). DIR ist das pdflib install directory, default ist `/usr/local`.

PHP 4: Inkludiert pdflib 3.x/4.x Unterstützung. DIR ist die pdflib install location, default ist `/usr/local`.

PHP 4 and PDFlib 3.x/4.x benötigen verfügbare JPEG und TIFF Bibliotheken. Wenn Sie PDFlib Unterstützung kompilieren, verwenden Sie die --with-jpeg-dir und --with-tiff-dir Konfigurationsoptionen. Sie können zusätzlich auch die Optionen --with-png-dir und --with-zlib-dir verwenden, um PNG und Zlib Unterstützung in die PDFlib Extension zu kompilieren.

`--enable-shared-pdflib`

PHP 3, PHP 4: Aktiviert pdflib als shared library.

`--with-readline[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert readline Unterstützung. DIR ist das readline install directory.

`--with-regex=TYPE`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: regex Bibliothekstypen: system, apache, php

`--with-servlet[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert servlet Unterstützung. DIR ist das base install directory für das JSDK. Dieses SAPI verlangt, dass die Java Extension als ein shared dl erstellt wird.

`--with-ming`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Flash 4 Unterstützung, mit Ming

`--with-swf[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert swf Unterstützung

`--with-system-regex`

PHP 3: Benutzt die bundled regex Bibliothek nicht

PHP 4: (nicht empfohlen) Verwenden der system regex Bibliothek

`--with-tsrm-pth[=pth-config]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Verwendet GNU Pth.

`--with-tsrm-pthreads`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Verwendet POSIX threads (default)

--with-x

PHP 3: Verwendet das X Window System

PHP 4: Ist in PHP 4 nicht verfügbar

--with-bz2[=DIR]

PHP 4: Inkludiert Unterstützung von bzip2. DIR ist das bzip2 Installationsverzeichnis.

--with-zlib-dir[=DIR]

PHP 3: zlib dir für pdflib 2.0 oder inkludiere zlib Unterstützung

PHP 4: zlib dir für pdflib 3.x/4.x oder inkludiere zlib Unterstützung

--with-zlib[=DIR]

PHP 3, PHP 4: Inkludiert zlib Unterstützung (benötigt zlib >= 1.0.9). DIR ist das zlib Installationsverzeichnis, default ist /usr.

--with-zip[=DIR]

PHP 4: Inkludiert zip Unterstützung (benötigt zziplib >= 0.10.6). DIR ist das zziplib Installationsverzeichnis, default ist /usr/local.

Die letzte Version von zziplib finden Sie auf http://zziplib.sourceforge.net/.

--without-pcre-regex

PHP 3: Ohne Perl Compatible Regular Expressions Unterstützung

PHP 4: Ohne Perl Compatible Regular Expressions Unterstützung. Verwenden Sie --with-pcre-regex=DIR, um mit DIR anzugeben, wo sich die Include- und Bibliothekendateien von PCRE befinden, wenn Sie die bundled Bibliothek nicht verwenden.

--without-posix

PHP 3: Ohne POSIX Funktionen

PHP 4: Ist in PHP 4 nicht verfügbar; verwenden Sie stattdessen --disable-posix.

Networking

--with-curl[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert CURL Unterstützung

`--enable-ftp`

PHP 3: Ist nicht verfügbar; verwenden Sie stattdessen --with-ftp.

PHP 4: Inkludiert FTP Unterstützung

`--with-ftp`

PHP 3: Inkludiert FTP Unterstützung.

PHP 4: Ist nicht verfügbar; verwenden Sie stattdessen --enable-ftp.

`--disable-url-fopen-wrapper`

PHP 3, PHP 4: Verhindere den URL-aware fopen wrapper, welcher den Zugriff auf Dateien über http oder ftp ermöglicht.

Warnung

Diese Option ist nur für PHP Versionen bis 4.0.3 verfügbar, aktuellere Versionen bieten den INI Parameter `allow_url_fopen`, damit Sie sich in diesem Punkt nicht schon bei der Kompilation festlegen müssen.

`--with-mod-dav=DIR`

PHP 3, PHP 4: Inkludiert DAV Unterstützung über Apache's mod_dav, DIR ist mod_dav's Installationsverzeichnis (Nur Apache Modulversion!)

`--with-openssl[=DIR]`

PHP 3, PHP 4: Inkludiert OpenSSL Unterstützung in SNMP.

`--with-snmp[=DIR]`

PHP 3, PHP 4: Inkludiert SNMP Unterstützung. DIR ist das SNMP base install directory, default ist die Suche in einigen für die snmp Installation üblichen Verzeichnissen. Setzen Sie DIR auf shared um als ein dl zu builden, oder shared,DIR um als ein dl zu builden und DIR anzugeben.

`--enable-ucd-snmp-hack`

PHP 3, PHP 4: Aktiviert UCD SNMP hack

`--enable-sockets`

PHP 3: Ist in in PHP 3 nicht verfügbar

PHP 4: Aktiviert sockets Unterstützung

`--with-yaz[=DIR]`

PHP 3: Ist in in PHP 3 nicht verfügbar

PHP 4: Inkludiert YAZ Unterstützung (ANSI/NISO Z39.50). DIR ist das YAZ bin install directory

--enable-yp

PHP 3: Ist nicht verfügbar, verwenden Sie stattdessen --with-yp.

PHP 4: Inkludiert YP Unterstützung

--with-yp

PHP 3: Inkludiert YP Unterstützung

PHP 4: Ist nicht verfügbar, verwenden Sie stattdessen --enable-yp.

--with-mnogosearch

PHP 3, PHP 4: Inkludiert mnoGoSearch Unterstützung.

PHP Verhalten

--enable-magic-quotes

PHP 3, PHP 4: Aktiviert magic quotes standardmäßig.

--disable-short-tags

PHP 3, PHP 4: Verhindert den short-form <? start tag standardmäßig.

--enable-track-vars

PHP 3: Aktiviert GET/POST/Cookie track variables standardmäßig.

PHP 4: Ist in PHP 4 nicht verfügbar; seit PHP 4.0.2 ist track_vars immer on.

Server

--with-aolserver-src=DIR

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Geben Sie den Pfad zur AOLserver source distribution an

--with-aolserver=DIR

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Geben Sie Pfad zu dem installierten AOLserver an

`--with-apache[=DIR]`

PHP 3, PHP 4: Erstelle ein Apache Modul. DIR ist das top-level Apache build directory, default ist /usr/local/etc/httpd.

`--with-apxs[=FILE]`

PHP 3, PHP 4: Erstellt shared Apache Modul. FILE ist der optionale Pfadname zu dem Apache apxs tool; default ist apxs.

`--enable-versioning`

PHP 3: Nutzt die Vorteile des Versionings und Scopings von Solaris 2.x und Linux

PHP 4: Exportiere nur die benötigten Symbole. Siehe INSTALL für mehr Informationen

`--with-caudium[=DIR]`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Erstellt PHP als ein Pike Modul zur Verwendung mit dem Caudium Webserver. DIR ist das Caudium base directory. Ist kein Directory angegeben, wird $prefix/caudium/server angenommen. Das Präfix wird von der Option --prefix bestimmt, default ist /usr/local.

`--with-fhttpd[=DIR]`

PHP 3, PHP 4: Erstellt das fhttpd Modul. DIR ist das fhttpd sources Directory, default ist /usr/local/src/fhttpd.

`--with-nsapi=DIR`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Geben Sie den Pfad zu dem installierten Netscape an.

`--with-phttpd=DIR`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4:

`--with-pi3web=DIR`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Erstellt PHP als ein Modul für Pi3Web.

`--with-roxen=DIR`

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Erstellt PHP als ein Pike Modul. DIR ist das base Roxen directory, normalerweise /usr/local/roxen/server.

--enable-roxen-zts

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Erstellt das Roxen Modul unter Verwendung der Zend Thread Safety.

--with-thttpd=SRCDIR

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4:

--with-zeus=DIR

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Erstellt PHP als ISAPI Modul für Zeus.

Text and Sprache

--with-aspell[=DIR]

PHP 3, PHP 4: Inkludiert ASPELL Unterstützung.

--with-gettext[=DIR]

PHP 3, PHP 4: Inkludiert GNU gettext Unterstützung. DIR ist das gettext Installationsverzeichnis, default ist /usr/local

--with-iconv[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar.

PHP 4: Inkludiert iconv Unterstützung.

--with-pspell[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar.

PHP 4: Inkludiert PSPELL Unterstützung.

--with-recode[=DIR]

PHP 3: Inkludiert GNU recode Unterstützung.

PHP 4: Inkludiert recode Unterstützung. DIR ist das recode Installationsverzeichnis.

--enable-shmop

PHP 3, PHP 4 : Aktiviert shmop Unterstützung.

XML

--with-dom[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert DOM Unterstützung (benötigt libxml >= 2.0). DIR ist das libxml Installationsverzeichnis, default ist `/usr`

--enable-sablot-errors-descriptive

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert Descriptive errors

--with-sablot[=DIR]

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Inkludiert Sablotron Unterstützung

--enable-wddx

PHP 3: Ist in PHP 3 nicht verfügbar

PHP 4: Aktiviert WDDX Unterstützung

--disable-xml

PHP 3: Ist in PHP 3 nicht verfügbar; XML Funktionalität ist nicht standardmäßig inkludiert. Verwenden Sie --with-xml, um sie zu aktivieren.

PHP 4: Verhindert XML Unterstützung mittels der bundled expat Bibliothek

--with-xml

PHP 3: Inkludiert XML Unterstützung

PHP 4: Ist nicht verfügbar; XML Unterstützung ist standardmäßig integriert. Verwenden Sie --disable-xml, um sie zu deaktivieren.

Installation auf Windows Systemen

Dieser Teil behandelt Windows 95/98/Me und Windows NT/2000/XP (Das Setup auf XP ist dzt. noch nicht getestet). Erwarten Sie nicht, dass PHP auf 16 Bit-Plattformen, wie z.B. Windows 3.1 läuft. Manchmal sprechen wir bei den unterstützten Plattformen von Win32.

Es gibt zwei Wege PHP für Windows zu installieren: entweder manuell, oder mittels dem InstallShield Installer.

Wenn Sie Microsoft Visual Studio besitzen, können Sie auch den Source Code von PHP selbst kompilieren.

Haben Sie PHP erst einmal auf Ihrem Windows System installiert, können Sie als Erweiterung der Funktionalität auch verschiedene Erweiterungen laden.

Windows InstallShield

Der von der Downloadseite auf http://www.php.net/ erhältliche Windows PHP Installer installiert die CGI Version von PHP, und konfiguriert auch die Webserver IIS, PWS, und Xitami. Beachten Sie aber auch, dass der InstallShield Installer zwar eine einfache Art der Installation von PHP ist, gleichzeitig aber auch in verschiedenen Aspekten eingeschränkt ist (z.B. wird ein automatisches Setup von Erweiterungen nicht unterstützt).

Installieren Sie Ihren gewünschten HTTP Server auf Ihrem System und stellen Sie sicher, dass er läuft.

Starten Sie nun das Installationsprogramm, und folgen Sie den Instruktionen des Installation Wizards. Es werden zwei Installationsarten unterstützt: Standard bietet zweckmäßige Standardeinstellungen wo dies möglich ist, und Advanced (erweitert), welche die gewünschten Einstellungen während der Installation abfragt.

Der Installation Wizard sammelt genug Informationen, um die `php.ini` zu erstellen, und den Webserver für den Einsatz von PHP zu konfigurieren. Für IIS und auch PWS auf Windows NT Workstation wird eine Liste mit den Knoten auf dem Server, sowie eine Liste mit den Script Mappings angezeigt aus welcher Sie die gewünschten Knoten auswählen können, auf denen PHP in die Script Mappings eingetragen werden soll.

Wenn die Installation fertig ist informiert Sie der Installer, ob Sie Ihr System oder den Server neu starten sollen, oder ob Sie einfach mit der Nutzung von PHP beginnen können.

Warnung

Seien Sie sich darüber im Klaren, dass dieses Setup von PHP nicht sicher ist. Wenn Sie ein sicheres Setup von PHP möchten, Setzen Sie PHP manuell auf, und wählen jede Option sorgfältig aus. Dieses automatische Setup bietet Ihnen eine sofort lauffähige PHP Installation, ist jedoch nicht für den Einsatz auf Online-Servern gedacht.

Manuelle Installationsschritte

Diese Installationsanleitung hilft Ihnen, PHP auf Ihrem Windows Webserver manuell zu installieren und zu konfigurieren. Laden Sie sich hierfür bitte die gezippte Distribution von der Downloadseite http://www.php.net/ herunter. Die Originalversion dieser Anleitung wurde von Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us) verfasst, welche Sie auf http://www.umesd.k12.or.us/php/win32install.html finden.

Diese Anleitung unterstützt Sie bei der manuellen Installation für:

- Personal Web Server 3 und 4 oder neuer
- Internet Information Server 3 und 4 oder neuer
- Apache 1.3.x

- OmniHTTPd 2.0b1 und höher
- Oreilly Website Pro
- Xitami
- Netscape Enterprise Server, iPlanet

PHP 4 für Windows gibt es in zwei Varianten - ein CGI Modul (php.exe), und verschiedene SAPI Module (z.B. php4isapi.dll). Letztere ist neu in PHP 4, und bietet eine signifikant höhere Performance, sowie neue Funktionalitäten.

Warnung

Beachten Sie bitte, dass die Qualität der SAPI Module für den Einsatz auf Produktionsservern dzt. als noch *NICHT* hoch genug eingestuft wird. Besonders beim ISAPI Modul können speziell auf älteren Plattformen als W2k ernsthafte Stabilitätsprobleme auftreten - Sie könnten z.B. einige Server 500 Errors erleben, und am Ausfall anderer Servermodule wie ASP leiden. Sie wurden gewarnt!

Der Grund dafür ist, dass die PHP SAPI Module die Thread-safe Version des PHP Codes nutzen. Diese ist neu in PHP 4, und wurde noch nicht eingehend genug getestet und belastet, um als völlig stabil angesehen werden zu können, und es sind dzt. auch ein paar Fehler bekannt. Andererseits haben einige Leute über sehr gute Resultate mit den SAPI Modulen berichtet, auch wenn wir über niemanden informiert sind, der sie auf Produktionsservern einsetzt. Kurz gesagt - Ihr Nutzen kann variieren; Wenn Sie absolute Stabilität benötigen, tauschen Sie die Performance der SAPI Module gegen die Stabilität des CGI Moduls.

Wenn Sie eines der SAPI Module unter Windows 95 einsetzen möchten, holen Sie sich das DCOM Update von den Microsoft DCOM Seiten (http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe). Für das ISAPI Modul wird ein ISAPI 4.0 kompatibler Webserver benötigt (getestet wurden IIS 4.0, PWS 4.0 und IIS 5.0). IIS 3.0 wird *NICHT* unterstützt. Wenn Sie gediegene PHP Unterstützung wünschen, sollten Sie In diesem Fall den Windows NT 4.0 Option Pack mit IIS 4.0 downloaden.

Die folgenden Schritte sollten vor den serverspezifischen Instruktionen an allen Installationen durchgeführt werden.

- Entpacken Sie die in der Distribution enthaltenen Dateien, in ein Verzeichnis Ihrer Wahl. Wir empfehlen "c:\php\".
- Stellen Sie sicher, dass die von PHP verwendeten DLLs gefunden werden können. Welche DLLs benötigt werden, hängt von dem verwendeten Webserver, und ob Sie PHP als CGI- oder Server-Modul einsetzen. `php4ts.dll` wird immer verwendet. Wenn Sie ein Server-Modul benutzen (z.B. ISAPI oder Apache), dann brauchen Sie die entsprechende DLL des `sapi` Verzeichnisses. Wenn Sie PHP extension DLLs benutzen, benötigen Sie diese ebenfalls. Um sicherzugehen, dass diese auch gefunden werden, können Sie sie entweder in das Systemverzeichnis (z.B. `winnt/system32` oder `windows/system`) kopieren, oder Sie können sie in das gleiche Verzeichnis stellen, in welchem sich auch die von Ihrem Webserver benutzte EXE- bzw. DLL-Datei befindet (z.B. php.exe, php4apache.dll).

  Die ’php.exe’, die SAPI Module, und einige Erweiterungen benötigen externe DLLs. Stellen Sie sicher, dass diese DLLs der Distribution in einem Verzeichnis liegen, welches im Windows PATH

liegt. Der einfachste Weg dazu ist, die nachstehenden Dateien in Ihr Systemverzeichnis zu kopieren. Dieses heißt gewöhnlich:
c:\windows\system auf Windows 9x/Me
c:\winnt\system32 auf Windows NT/2000
Die zu kopierenden Dateien sind:
'php4ts.dll'. Wenn sie dort bereits existiert, überschreiben Sie sie.
Die Dateien in dem Verzeichnis 'dlls' Ihrer Distribution. Sollten diese Dateien bereits auf Ihrem System installiert se

Holen Sie sich die letzte Version der Microsoft Data Access Components (MDAC) für Ihre Plattform, speziell wenn Sie Windows 9x/NT4 benutzen. MDAC finden Sie unter http://www.microsoft.com/data/.

- Kopieren Sie die gewünschte ini Datei (siehe unten) in Ihr '%WINDOWS%' Verzeichnis auf Windows 9x/ME, oder in Ihr '%SYSTEMROOT%' Verzeichnis unter Windows NT oder Windows 2000, und benennen Sie sie in `php.ini` um. Ihr '%WINDOWS%' oder '%SYSTEMROOT%' Verzeichnis ist normalerweise:
c:\windows für Windows 9x/ME
c:\winnt oder c:\winnt40 für NT/2000 Server

  In der Zip-Datei befinden sich zwei ini Dateien: `php.ini-dist` und `php.ini-optimized`. Wir empfehlen, die `php.ini-optimized` zu verwenden, da in dieser Datei die Default-Einstellungen für Performance und Sicherheit optimiert sind. Am besten ist es, all die ini Einstellungen selbst manuell vorzunehmen. Wenn Sie die bestmögliche Sicherheit erreichen wollen ist dies der Weg für Sie, auch wenn PHP mit den Standardeinstellungen gut läuft.

- Editieren Sie nun Ihre Datei 'php.ini':
  - Tragen Sie bei 'extension_dir' Ihr PHP Installationsverzeichnis oder den Pfad zu Ihren 'php_*.dll' Dateien ein, z.B.: c:\php\extensions
  - Wenn Sie OmniHTTPd benutzen, überspringen Sie diesen Schritt. Geben Sie bei 'doc_root' Dokumentenverzeichnis des Webservers an. z.B.: c:\apache\htdocs oder c:\webroot
  - Wählen Sie nun die Erweiterungen (Module) aus, die Sie benutzen möchten, wenn PHP startet. Lesen Sie in Windows Erweiterungen wie Sie eine aktivieren können, und welche bereits inkludiert sind. Bei einer Neuinstallation ist es ratsam, erst PHP zu installieren und zu testen, bevor Sie Erweiterungen in der `php.ini` aktivieren.
  - Auf PWS und IIS können Sie auch die `browscap.ini` auf 'c:\windows\system\inetsrv\browscap.ini' unter Windows 9x/Me, und 'c:\winnt\system32\inetsrv\browscap.ini' unter NT/2000 Server setzen.
  - Beachten Sie, dass das `mibs` Verzeichnis in der Windows Distribution Dateien für die SNMP Unterstützung enthält. Dieses Verzeichnis sollte nach `DRIVE:\usr\mibs` verschoben werden (`DRIVE` ist das Laufwerk, auf dem PHP installiert ist).

Den Source kompilieren

Bevor wir beginnen lohnt es sich, folgende Frage zu beantworten: "Warum ist das Kompilieren unter Windows so schwer?" Zwei Gründe drängen sich auf:

1. Windows genießt (noch) keine große Gemeinschaft von Entwicklern welche auch bereit sind, deren Sourcecode frei zur Verfügung zu stellen. Als Ergebnis wurden die nötigen Investitionen

in die für solche Entwicklungen nötige Infrastruktur nicht getätigt. Das meiste was verfügbar ist, wurde durch die Portierung der nötigen Werkzeuge von Unix möglich gemacht. Seien Sie deshalb nicht überrascht, wenn manches aus diesem Erbe von Zeit zu Zeit durchscheint.

2. Fast alle der folgenden Instruktionen sind von der Art "Einstellen und vergessen". Deshalb lehnen Sie sich zurück, und folgen den nachstehenden Instruktionen so gewissenhaft Sie können.

Vorbereitungen

Bevor Sie beginnen können, müssen Sie einiges downloaden...

- Erst einmal benötigen Sie den Cygwin Toolkit von einem Cygwin (http://sources.redhat.com/cygwin/download.html) Mirror in Ihrer Nähe. Cygwin bietet die meisten der populären GNU Utilities, welche während der Kompilation benötigt werden.
- Laden Sie den Rest der benötigten Tools von der PHP Seite http://www.php.net/extra/win32build.zip herunter.
- Laden Sie sich auch den Sourcecode für die von PHP verwendete DNS Namensauflösung von http://www.php.net/extra/bindlib_w32.zip herunter. Dieser ersetzt die in `win32build.zip` enthaltene Bibliothek `resolv.lib`.
- Sollten Sie noch kein Tool zur dekomprimierung besitzen, so bekommen Sie eine freie Version von InfoZip (http://www.cdrom.com/pub/infozip/UnZip.html).

Als letztes benötigen Sie natürlich den Sourcecode von PHP 4 selbst. Sie bekommen die aktuellste Entwicklerversion unter Verwendung des anonymen CVS (http://www.php.net/anoncvs.php). Wenn Sie einen Snapshot (http://snaps.php.net/) oder einen Source (http://www.php.net/downloads.php) Tarball downloaden, müssen Sie ihn nicht nur erst dekomprimieren, sondern auch in den `*.dsp` und `*.dsw` Dateien die einfachen Linefeeds in crlf konvertieren, bevor diese von Microsoft Visual C++ geladen werden.

> **Anmerkung:** Stellen Sie die Verzeichnisse `Zend` und `TSRM` in das `php4` Verzeichnis, damit die Projekte während des Kompilierens gefunden werden.

Alles zusammenfügen

- Folgen Sie den Anweisungen zur Installation des von Ihnen gewählten Dekomprimierungstools.
- Führen Sie `setup.exe` aus, und folgen Sie den Installationsanweisungen. Wenn Sie einen anderen Pfad als `c:\cygnus` wählen, setzen Sie bitte die Cygwin Environmentvariable. Unter Windows 95/98 kann dies mittels einer Zeile in der autoexec.bat erfolgen. Unter Windows NT gehen Sie auf Systemsteuerung => System und wählen den Tab Umgebungsvariablen aus.

> **Warnung**
>
> Erstellen Sie ein temporäres Verzeichnis für Cygwin, sonst werden einige Kommandos (speziell bison) nicht korrekt ausgeführt. Unter Windows 95/98, **`mkdir C:\TMP`**. Für Windows NT, **`mkdir %SystemDrive%\tmp`**.

- Erstellen Sie ein Verzeichnis, und dekomprimieren Sie darin `win32build.zip`.
- Starten Sie Microsoft Visual C++, und wählen im Menü Extras => Optionen => Verzeichnisse. Wechseln Sie nacheinander zu "Ausführbare Dateien", "Include Dateien" und "Bibliothek Dateien", und vergewissern Sie sich, dass `cygwin\bin`, `win32build\include`, und `win32build\lib` in der jeweiligen Liste eingetragen sind (Um einen Eintrag hinzuzufügen, wählen Sie eine leere Zeile am Ende der Liste aus, und beginnen einzutippen). Typische Einträge sehen wie folgt aus:
  - `c:\cygnus\bin`
  - `c:\php-win32build\include`
  - `c:\php-win32build\lib`

  Drücken Sie OK, und verlassen Sie Visual C++.

- Erstellen Sie ein anderes Verzeichnis, und dekomprimieren Sie darin `bindlib_w32.zip`. Entscheiden Sie sich, ob Sie die Debug-Symbole verfügbar haben wollen (bindlib - Win32 Debug) oder nicht (bindlib - Win32 Release). Erstellen die geeignete Konfiguration:
  - Wenn Sie das GUI benutzen: Starten Sie VC++, selektieren Sie Datei => Arbeitsbereich öffnen, und wählen Sie bindlib aus. Dann selektieren Sie Build => Aktive Konfiguration, und wählen die gewünschte Konfiguration aus. Letztendlich wählen Sie Erstellen => Rebuild All.
  - Wenn Sie die Kommandozeile benutzen stellen Sie sicher, dass die C++ Environmentvariablen registriert sind, oder starten Sie **vcvars.bat**, und führen eines der folgenden Kommandos aus:
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`**
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`**

  - Nun sollten Sie entweder in Ihrem `Debug` oder `Release` Verzeichnis eine verwendbare `resolv.lib` haben. Kopieren Sie diese Datei in Ihr `win32build\lib` Verzeichnis, und überschreiben damit die dort bereits vorhandene Datei mit dem gleichen Namen.

Kompilieren

Der einfachste Weg zu beginnen, ist die standalone/CGI Version zu erstellen.

- Wenn Sie das GUI benutzen: Starten Sie VC++, selektieren Sie Datei => Arbeitsbereich öffnen, und wählen Sie php4ts aus. Dann selektieren Sie Erstellen => Setze aktuelle Konfiguration, und wählen die gewünschte Konfiguration aus. Letztendlich wählen Sie Erstellen => Rebuild All.
- Wenn Sie die Kommandozeile benutzen stellen Sie sicher, dass die C++ Environmentvariablen registriert sind, oder starten Sie **vcvars.bat**, und führen eines der folgenden Kommandos aus:
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`**
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`**
  - Nun sollten Sie entweder in Ihrem `Debug_TS` oder `Release_TS` Verzeichnis eine verwendbare `php.exe` haben.

Wiederholen Sie die obigen Schritte mit `php4isapi.dsp` (welches sich in `sapi\isapi` befindet), um das für eine Integration mit Microsoft IIS nötige ISAPI-Modul zu erstellen.

Installation von Windows Erweiterungen

Sind PHP und ein Webserver einmal auf Windows installiert, möchten Sie vielleicht einige Erweiterungen für zusätzliche Funktionalität installieren. Die folgende Tabelle beschreibt einige der verfügbaren Erweiterungen. Sie können Erweiterungen auswählen, welche beim Start von PHP geladen werden, indem Sie das Kommentarzeichen (';') in den entsprechenden 'extension=php_*.dll' Zeilen in der `php.ini` entfernen. Sie können Module mittels dl() auch dynamisch laden.

Die DLLs für PHP Erweiterungen enthalten in PHP 4 als Präfix 'php_' (und 'php3_' in PHP 3), um Verwechslungen zwischen PHP Erweiterungen und deren unterstützenden Bibliotheken zu vermeiden.

Anmerkung: In PHP 4.0.6 ist die Unterstützung für BCMath, Calendar, COM, FTP, MySQL, ODBC, PCRE, Session, WDDX und XML *inkludiert*. Um diese Funktionen zu benutzen, brauchen Sie keine zusätzlichen Erweiterungen zu laden. Eine Liste mit den bereits inkludierten Modulen befindet sich in der `README.txt` oder `install.txt` Ihrer Distribution.

Anmerkung: Einige Erweiterungen benötigen externe DLLs, wovon bereits einige im Verzeichnis 'dlls' der PHP Distribution enthalten sind. Manche Erweiterungen benötigen jedoch DLLs, welche nicht in der PHP Distribution enthalten sind, wie z.B. Oracle (php_oci8.dll).

Kopieren Sie die DLLs vom Verzeichnis 'dlls' in Ihren Windows PATH, gute Plätze sind:

c:\windows\system für Windows 9x/ME
c:\winnt\system32 für Windows NT/2000

Sollten diese Dateien bereits auf Ihrem System installiert sein, überschreiben Sie diese nur, wenn etwas nicht korrekt läuft (Vor dem Überschreiben empfehlen wir ein Backup, oder verschieben diese Dateien in ein anderes Verzeichnis - nur für den Fall dass etwas schiefgeht).

Tabelle 2-1. PHP Erweiterungen

Erweiterung	Beschreibung	Anmerkungen
php_bz2.dll	bzip2 compression functions	Keine
php_calendar.dll	Kalender Umwandlungsfunktionen	Ist seit PHP 4.0.3 integriert
php_cpdf.dll	ClibPDF Funktionen	Keine
php3_crypt.dll	Verschlüsselungsfunktionen	Keine
php_ctype.dll	ctype Funktionen	Keine
php_curl.dll	CURL, Client URL Bibliotheksfunktionen	Benötigt libeay32.dll, ssleay32.dll (bundled)
php_cybercash.dll	Cybercash Zahlungssystem Funktionen	Keine

Erweiterung	Beschreibung	Anmerkungen
php_db.dll	DBM Funktionen	Nicht empfohlen. Verwenden Sie stattdessen DBA (php_dba.dll)
php_dba.dll	DBA Datenbank (dbm-style) Abstraktions-Funktionen	Keine
php_dbase.dll	dBase Funktionen	Keine
php3_dbm.dll	Berkeley DB2 Bibliothek	Keine
php_domxml.dll	DOM XML Funktionen	Benötigt libxml2.dll (bundled)
php_dotnet.dll	.NET Funktionen	Keine
php_exif.dll	Liest EXIF Headers einer JPEG-Grafik	Keine
php_fbsql.dll	FrontBase Funktionen	Keine
php_fdf.dll	FDF: Forms Data Format Funktionen	Benötigt fdftk.dll (bundled)
php_filepro.dll	filePro Funktionen	Read-only Zugriff
php_ftp.dll	FTP Funktionen	Ist seit PHP 4.0.3 integriert
php_gd.dll	GDGD Bibliotheksfunktionen zur Bildmanipulation	Keine
php_gettext.dll	Gettext Funktionen	Benötigt gnu_gettext.dll (bundled)
php_hyperwave.dll	HyperWave Funktionen	Keine
php_iconv.dll	ICONV Zeichensatzumwandlung	Benötigt iconv-1.3.dll (bundled)
php_ifx.dll	Informix Funktionen	Benötigt Informix Bibliotheken
php_iisfunc.dll	IIS Managementfunktionen	Keine
php_imap.dll	IMAP POP3 und NNTP Funktionen	PHP 3: php3_imap4r1.dll
php_ingres.dll	Ingres II Funktionen	Benötigt Ingres II Bibliotheken
php_interbase.dll	InterBase Funktionen	Benötigt gds32.dll (bundled)
php_java.dll	Java Erweiterung	Benötigt jvm.dll (bundled)
php_ldap.dll	LDAP Funktionen	Benötigt libsasl.dll (bundled)
php_mhash.dll	Mhash Funktionen	Keine
php_ming.dll	Ming Funktionen für Flash	Keine
php_msql.dll	mSQL Funktionen	Benötigt msql.dll (bundled)
php3_msql1.dll	mSQL 1 Client	Keine
php3_msql2.dll	mSQL 2 Client	Keine
php_mssql.dll	MSSQL Funktionen	Benötigt ntwdblib.dll (bundled)
php3_mysql.dll	MySQL Funktionen	in PHP 4 bereits integriert
php3_nsmail.dll	Netscape Mail Funktionen	Keine
php3_oci73.dll	Oracle Funktionen	Keine
php_oci8.dll	Oracle 8 Funktionen	Benötigt Oracle 8 Client Bibliotheken
php_openssl.dll	OpenSSL Funktionen	Benötigt libeay32.dll (bundled)

Erweiterung	Beschreibung	Anmerkungen
php_oracle.dll	Oracle Funktionen	Benötigt Oracle 7 Client Bibliotheken
php_pdf.dll	PDF Funktionen	Keine
php_pgsql.dll	PostgreSQL Funktionen	Keine
php_printer.dll	Printer Funktionen	Keine
php_sablot.dll	XSLT Funktionen	Benötigt sablot.dll (bundled)
php_snmp.dll	SNMP get and walk Funktionen	nur NT!
php_sybase_ct.dll	Sybase Funktionen	Benötigt Sybase Client Bibliotheken
php_yaz.dll	YAZ Funktionen	Keine
php_zlib.dll	ZLib Komprimierungsfunktionen	Keine

Server - Apache

Dieser Teil enthält Anmerkungen und Tipps zur Installation von Apache mit PHP, sowohl für die Unix als auch für die Windows Version.

Details der Installation von PHP mit Apache auf Unix

Aus der Kompletten Liste der Konfigurationsoptionen können Sie die weitere Argumente für **configure** in Zeile 8 wählen.

Beispiel 2-5. Installationsanweisungen (Apache Modulversion)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. for PHP 3: ./configure --activate-module=src/modules/php3/libphp3.a
    for PHP 4: ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install

  Statt diesem Schritt können Sie auch die httpd Binärdatei einfach über
  Ihre existierende Binärdatei kopieren. Stellen Sie jedoch sicher, dass
  der Server zuvor heruntergefahren wurde.

15. cd ../php-x.x.x
```

```
16. for PHP 3: cp php3.ini-dist /usr/local/lib/php3.ini
    for PHP 4: cp php.ini-dist /usr/local/lib/php.ini

  Sie können Ihre .ini Datei editieren, um PHP Optionen auszuwählen.
  Wenn Sie diese Datei an einer anderen Stelle haben möchten, verwenden
  Sie in Schritt 8 --with-config-file-path=/path

17. Editieren Sie Ihre httpd.conf oder srm.conf Datei, und fügen Sie
    hinzu:

     Für PHP 3:   AddType application/x-httpd-php3 .php3
     Für PHP 4:   AddType application/x-httpd-php .php

  Sie können jede gewünschte Dateinamenserweiterung wählen,  .php wird
  von uns einfach nur vorgeschlagen. Sie können selbst .html einbinden.

18. Verwenden Sie Ihre normale Prozedur, um den Apache Server zu starten.
    (Sie müssen den Server stoppen und wieder starten, und ihn nicht nur
    mittels einem HUP oder USR1 Signal zu einem Reload veranlassen).
```

Es gibt abhängig von Ihrer Apache Installation und Unix Variante viele mögliche Wege den Server zu stoppen und neu zu starten. Nachstehend finden Sie einige typische Zeilen für einen Server Restart auf verschiedenen apache/unix Installationen. Ersetzen Sie `/path/to/` mit dem Pfad zu diesen Applikationen auf Ihrem System.

```
1. Verschiedene Linux und SysV Varianten:
/etc/rc.d/init.d/httpd restart

2. Verwendung von apachectl Skripten:
/path/to/apachectl stop
/path/to/apachectl start

3. httpdctl und httpsdctl (Verwendet OpenSSL), ähnlich zu apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start

4. Verwenden von mod_ssl oder einem anderen SSL Server, vielleicht
möchten Sie manuell stoppen und wieder starten:
/path/to/apachectl stop
/path/to/apachectl startssl
```

Der Ort der apachectl und http(s)dctl Binaries ändert sich häufig. Wenn Ihr System über `locate`, `whereis` oder `which` Kommandos verfügt, sind diese beim Auffinden Ihrer Server contrl Programme hilfreich.

Verschiedene Beispiele zum Kompilieren von PHP für Apache:

```
./configure --with-apxs --with-pgsql
```

Dies erstellt eine `libphp4.so` Shared Library, welche in den Apache geladen wird, wenn eine LoadModule Zeile in der `httpd.conf` von Apache existiert. Die PostgreSQL Unterstützung ist in der `libphp4.so` Bibliothek eingelagert.

```
./configure --with-apxs --with-pgsql=shared
```

Dies erstellt ebenfalls eine `libphp4.so` Shared Library, jedoch wird zusätzlich auch eine `pgsql.so` Shared Library erstellt. Diese wird entweder aufgrund der Extension Option in der `php.ini`, oder durch die Verwendung der Funktion dl() in einem Skript von PHP geladen.

```
./configure --with-apache=/path/to/apache_source --with-pgsql
```

Dies erstellt eine `libmodphp4.a` Bibliothek, eine `mod_php4.c` und einige begleitende Dateien, und kopiert diese in das `src/modules/php4` Verzeichnis im Apache Source Tree. Dann kompilieren Sie Apache mit `--activate-module=src/modules/php4/libphp4.a`, und das Apache Build System erstellt `libphp4.a`, und linkt es statisch in die `httpd` Binary. Die PostgreSQL Unterstützung ist in dieser `httpd` Binary inkludiert, was in einer einzigen `httpd` Binary inkl. komplettem Apache und komplettem PHP resultiert.

```
./configure --with-apache=/path/to/apache_source --with-pgsql=shared
```

Wie vorher, außer dass die PostgreSQL Unterstützung nicht direkt in die `httpd` inkludiert, sondern eine `pgsql.so` Shared Library erstellt wird. Diese können Sie entweder mittels der Extension Option in der `php.ini`, oder direkt mittels dl() in PHP laden.

Bevor Sie sich für einen Weg PHP zu kompilieren entscheiden, sollten Sie die Vor- und Nachteile der jeweiligen Methode bedenken. Die Kompilation als verteiltes Objekt heißt, dass Sie Apache separat kompilieren können, und nicht für alles in PHP hinzugefügte bzw. geänderte neu kompilieren müssen. PHP direkt in Apache einzukompilieren (statische Methode) heißt, dass PHP schneller lädt, und schneller läuft. Weitere Informationen finden Sie auf der Apache webpage on DSO support (http://www.apache.org/docs/dso.html).

Installation von PHP auf Windows mit Apache 1.3.x

Es gibt zwei Arten, PHP mit Apache 1.3.x unter Windows aufzusetzen. Eine ist die Verwendung des CGi Moduls (php.exe), die andere ist der Einsatz der Apache Modul dll. In beiden Fällen müssen Sie den Apache Server stoppen, und Ihre `srm.conf` oder `httpd.conf` editieren, um Apache für die Zusammenarbeit mit PHP zu konfigurieren.

Obwohl einige Variationen in der Konfiguration von PHP unter Apache möglich sind, sollten diese auch für Neulinge einfach genug sein. Für weitere Konfigurationsmöglichkeiten konsultieren Sie bitte die Apache Dokumentation.

Wenn Sie die PHP Distribution wie im Teil Manuelle Installationsschritte beschrieben nach c:\php\ entpackt haben, müssen Sie die folgenden Zeilen in Ihrer Apache Konfigurationsdatei einfügen, um das CGI-Modul aufzusetzen:

- `ScriptAlias /php/ "c:/php/"`
- `AddType application/x-httpd-php .php .phtml`
- `Action application/x-httpd-php "/php/php.exe"`

Beachten Sie, dass die zweite Zeile der obigen Liste in den aktuellen Versionen von `httpd.conf` bereits enthalten, jedoch auskommentiert ist. Vergessen Sie nicht, nach der Änderung der Konfigurationsdatei den Server neu zu starten, z.B. mit `NET STOP APACHE`, gefolgt von `NET START APACHE` wenn Sie Apache als Windows Service nutzen, oder verwenden Sie Ihre Shortcuts.

Wenn Sie PHP als Apache Modul einsetzen möchten, kopieren Sie die `php4ts.dll` in Ihr windows/system (für Windows 9x/Me) oder winnt/system32 (für Windows NT/2000) Verzeichnis, und überschreiben die dort eventuell bereits existierende Datei. Dann sollten Sie die folgenden zwei Zeilen in Ihrer Apache conf Datei einfügen:

- `LoadModule php4_module c:/php/sapi/php4apache.dll`
- `AddType application/x-httpd-php .php .phtml`

> **Anmerkung:** Apache 1.3.22 for Windows hat in seiner Konfigurationsdatei (`httpd.conf-dist-win`) die `ClearModuleList` Direktive standardmäßig inkludiert. Wird diese Direktive verwendet, muss eine `AddModule mod_php4.c` in die `AddModule` Liste aufgenommen werden, da PHP sonst nicht als ein Apache Modul registriert wird.

Um das Source Code Highlighting Feature verwenden zu können, erstellen Sie einfach eine PHP Skriptdatei, und fügen diesen Code ein: `<?php show_source ("original_php_script.php"); ?>`. Ersetzen Sie `original_php_script.php` durch den Dateinamen der Datei, dessen Sourcecode Sie anzeigen lassen möchten (Das ist der einzige Weg, dies zu tun, da unter Windows kein Feature wie .phps existiert).

> **Anmerkung:** Auf Win-Apache müssen alle Backslashes "\" in einer Pfadangabe, wie z.B. "c:\directory\file.ext", in Slashes "/" umgewandelt werden, wie z.B. "c:/directory/file.ext".

Server - CGI/Commandline

Standardmäßig wird PHP als ein CGI Programm kompiliert. Dies erstellt einen Commandline Interpreter, welcher für CGI Processing oder für nicht webbezogenes PHP Scripting verwendet werden kann. Haben Sie einen Webserver im Einsatz für welchen PHP eine Modulunterstützung bietet, sollten sie aus Performancegründen dieses Modul verwenden. Die CGI Version ermöglicht Apache Usern, verschiedene PHP Seiten unter verschiedenen User-IDs laufen zu lassen. Bitte lesen Sie das Kapitel Sicherheit, wenn Sie PHP als CGI verwenden.

Testen

Wenn Sie PHP als CGI Programm erstellt haben, können Sie Ihren Build mittels **make test** testen, was generell eine gute Idee ist. Damit können Sie ein eventuelles Problem mit PHP auf Ihrer Plattform erkennen, bevor Sie später damit zu kämpfen haben.

Benchmarking

Haben Sie PHP 3 als CGI Programm erstellt, können Sie Ihren Build mit **make bench** bewerten. Beachten Sie, dass wenn Safe Mode standardmäßig aktiviert ist, der Benchmark nicht fertiggestellt werden kann, sollte dieser länger als die erlaubten 30 Sekunden dauern. Der Grund dafür liegt in der Tatsache, dass set_time_limit() in Safe Mode nicht benutzt werden kann. Verwenden Sie die Konfigurationsoption max_execution_time, um diese Zeit für Ihre eigenen Skripte einzustellen. **make bench** ignoriert die Konfigurationsdatei.

Anmerkung: make bench ist nur in PHP 3 verfügbar.

Server - fhttpd

Um PHP als ein fhttpd Modul zu erstellen, beantworten Sie die Frage "Build as an fhttpd module?" mit "yes" (die Konfigurationsoption --with-fhttpd=*DIR*), und geben das Basisverzeichnis des Source an. Das Standardverzeichnis ist `/usr/local/src/fhttpd`. Wenn Sie fhttpd benutzen, bietet das Kompilieren als Modul eine bessere Performance, mehr Kontrolle, sowie die Möglichkeit zur Remote Execution.

Server - Caudium

PHP 4 kann als Pike Modul für den Caudium Webserver erstellt werden. Dies wird mit PHP 3 nicht unterstützt. Folgen Sie einfach den nachstehenden Instruktionen, um PHP 4 für Caudium zu installieren.

Beispiel 2-6. Caudium Installationsanweisungen

```
1.  Installieren Sie Caudium, bevor Sie mit der Installation von PHP 4
    beginnen. Damit PHP 4 einwandfrei arbeitet, benötigen Sie Pike
    7.0.268 oder neuer. In diesem Beispiel nehmen wir an, dass Caudium
    in /opt/caudium/server/ installiert ist.
2.  Wechseln Sie in das Verzeichnis php-x.y.z (wobei x.y.z die
    Versionsnummer darstellt)
3.  ./configure --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Starten Sie Caudium neu, sollte er gerade laufen.
7.  Öffnen Sie die grafische Konfigurationsoberfläche, und wechseln Sie
    zu dem virtuellen Server, welcher PHP 4 Unterstützung bieten soll.
8.  Klicken Sie auf Add Module and locate, und fügen das PHP 4 Script
    Support Modul hinzu.
```

```
9.  Sollte die Dokumentation meinen 'PHP 4 interpreter isn't available',
    vergewissern Sie sich, dass der Server neu gestartet wurde. Prüfen
    Sie /opt/caudium/logs/debug/default.1 auf Fehler in Bezug auf
    PHP4.so. Vergewissern Sie sich ebenfalls, dass
    caudium/server/lib/[pike-version]/PHP4.so
    vorhanden ist.
10. Konfigurieren Sie das PHP Script Support Module wenn nötig.
```

Sie können Ihr Caudium Modul natürlich mit Unterstützung der verschiedenen in PHP 4 verfügbaren Erweiterungen kompilieren. Die Komplette Liste der Konfigurationsoptionen bietet hierfür eine gute Übersicht.

Anmerkung: Wenn Sie PHP 4 mit MySQL Unterstützung kompilieren müssen Sie sicherstellen, dass der normale MySQL Client Code verwendet wird. Die tun Sie, indem Sie ein MySQL Installationsverzeichnis mittels der --with-mysql Option spezifizieren.

Servers-IIS/PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server). Installing PHP for PWS/IIS 3, PWS 4 or newer and IIS 4 or newer versions.

Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the REG file included with the distribution (pws-php4cgi.reg). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

Warnung

These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- On the edit menu select: `New->String Value`.
- Type in the extension you wish to use for your php scripts. For example `.php`
- Double click on the new string value and enter the path to `php.exe` in the value data field. ex: `c:\php\php.exe`.
- Repeat these steps for each extension you wish to associate with PHP scripts.

The following steps do not affect the web server installation and only apply if you want your php scripts to be executed when they are run from the command line (ex. run `c:\myscripts\test.php`) or by double clicking on them in a directory viewer window. You may wish to skip these steps as you might prefer the PHP files to load into a text editor when you double click on them.

- Navigate to: `HKEY_CLASSES_ROOT`
- On the edit menu select: `New->Key`.
- Name the key to the extension you setup in the previous section. ex: `.php`
- Highlight the new key and in the right side pane, double click the "default value" and enter `phpfile`.
- Repeat the last step for each extension you set up in the previous section.
- Now create another `New->Key` under `HKEY_CLASSES_ROOT` and name it `phpfile`.
- Highlight the new key `phpfile` and in the right side pane, double click the "default value" and enter `PHP Script`.
- Right click on the `phpfile` key and select `New->Key`, name it `Shell`.
- Right click on the `Shell` key and select `New->Key`, name it `open`.
- Right click on the `open` key and select `New->Key`, name it `command`.
- Highlight the new key `command` and in the right side pane, double click the "default value" and enter the path to `php.exe`. ex: `c:\php\php.exe -q %1`. (don't forget the `%1`).
- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool (http://www.genusa.com/iis/iiscfg.html) from Steven Genusa to configure their script maps.

Windows and PWS 4 or newer

When installing PHP on Windows with PWS 4 or newer version, you have two options. One to set up the PHP CGI binary, the other is to use the ISAPI module dll.

If you choose the CGI binary, do the following:

- Edit the enclosed `pws-php4cgi.reg` file (look into the sapi dir) to reflect the location of your php.exe. Forward slashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\ScriptMap] ".php"="c:\\php\\php.exe"`
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

If you choose the ISAPI module, do the following:

- Edit the enclosed `pws-php4isapi.reg` file (look into the sapi dir) to reflect the location of your php4isapi.dll. Forward slashes should be escaped, for example:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script
Map] ".php"="c:\\php\\sapi\\php4isapi.dll"
```

- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

Windows NT/2000 and IIS 4 or newer

To install PHP on an NT/2000 Server running IIS 4 or newer, follow these instructions. You have two options to set up PHP, using the CGI binary (php.exe) or with the ISAPI module.

In either case, you need to start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

- Under 'Home Directory', 'Virtual Directory', or 'Directory', click on the 'Configuration' button, and then enter the App Mappings tab.
- Click Add, and in the Executable box, type: `c:\php\php.exe` (assuming that you have unziped PHP in c:\php\).
- In the Extension box, type the file name extension you want associated with PHP scripts. Leave 'Method exclusions' blank, and check the Script engine checkbox. You may also like to check the 'check that file exists' box - for a small performance penalty, IIS (or PWS) will check that the script file exists and sort out authentication before firing up php. This means that you will get sensible 404 style error messages instead of cgi errors complaing that php did not output any data.

  You must start over from the previous step for each extension you want associated with PHP scripts. `.php` and `.phtml` are common, although `.php3` may be required for legacy applications.

- Set up the appropriate security. (This is done in Internet Service Manager), and if your NT Server uses NTFS file system, add execute rights for I_USR_ to the directory that contains `php.exe`.

To use the ISAPI module, do the following:

- If you don't want to perform HTTP Authentication using PHP, you can (and should) skip this step. Under ISAPI Filters, add a new ISAPI filter. Use PHP as the filter name, and supply a path to the php4isapi.dll.
- Under 'Home Directory', click on the 'Configuration' button. Add a new entry to the Application Mappings. Use the path to the php4isapi.dll as the Executable, supply `.php` as the extension, leave Method exclusions blank, and check the Script engine checkbox.
- Stop IIS completely (NET STOP iisadmin)
- Start IIS again (NET START w3svc)

Servers-Netscape and iPlanet

This section contains notes and hints specific to Netscape and iPlanet installs of PHP, both for Sun Solaris and Windows versions.

You can find more information about setting up PHP for the Netscape Enterprise Server here: http://benoit.noss.free.fr/php/install-php4.html

Installing PHP with Netscape on Sun Solaris

To build PHP with NES or iPlanet web servers, enter the proper install directory for the --with-nsapi = *DIR* option. The default directory is usually `/opt/netscape/suitespot/`. Please also read `/php-xxx-version/sapi/nsapi/nsapi-readme.txt`.

Beispiel 2-7. Installation Example for Netscape Enterprise on Solaris

```
Instructions for Sun Solaris 2.6 with Netscape Enterprise Server 3.6
From: bhager@invacare.com

1. Install the following packages from www.sunfreeware.com or another
download site:

    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (if you want mysql support)
    tar-1.13 (GNU tar)

2. Make sure your path includes the proper directories
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH

3. gunzip php-x.x.x.tar.gz (if you have a .gz dist, otherwise go to 4)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x

6. For the following step, make sure /opt/netscape/suitespot/ is where
   your netscape server is installed. Otherwise, change to correct path:
    /configure --with-mysql=/usr/local/mysql --with-nsapi=/opt/netscape/suitespot/ -
-enable-track-vars --enable-libgcc
7. make
8. make install
```

After performing the base install and reading the appropriate readme file, you may need to performs some additional configuration steps.

Firstly you may need to add some paths to the LD_LIBRARY_PATH environment for Netscape to find all the shared libs. This can best done in the start script for your Netscape server. Windows users

can probably skip this step. The start script is often located in: `/path/to/server/https-servername/start`

You may also need to edit the configuration files that are located in:`/path/to/server/https-servername/config/`.

Beispiel 2-8. Configuration Example for Netscape Enterprise

```
Configuration Instructions for Netscape Enterprise Server
From: bhager@invacare.com

1. Add the following line to mime.types:
    type=magnus-internal/x-httpd-php exts=php

2. Add the following to obj.conf, shlib will vary depending on
    your OS, for unix it will be something like
    /opt/netscape/suitespot/bin/libphp4.so.

    You should place the following lines after mime types init.
    Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shli
    Init fn=php4_init errorString="Failed to initialize PHP!"

    <object name="default">
    .
    .
    .
    .#NOTE this next line should happen after all 'ObjectType' and before all 'AddLog' li
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>


    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>


    Authentication configuration

    PHP authentication cannot be used with any other authentication. ALL AU-
THENTICATION IS
    PASSED TO YOUR PHP SCRIPT. To configure PHP Authentication for the entire server, add
    the following line:

    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
    .
    </Object>

    To use PHP Authentication on a single directory, add the following:

    <Object ppath="d:\path\to\authenticated\dir\*">
```

```
AuthTrans fn=php4_auth_trans
</Object>
```

If you are running Netscape Enterprise 4.x, then you should use the following:

Beispiel 2-9. Configuration Example for Netscape Enterprise 4.x

```
Place these lines after the mime types init, and everything else is similar
to the example configuration above.
From: Graeme Hoose (GraemeHoose@BrightStation.com)

Init fn="load-modules" shlib="/path/to/server4/bin/libphp4.so" funcs="php4_init,php4_clos
Init fn="php4_init" LateInit="yes"
```

Installing PHP with Netscape on Windows

To Install PHP as CGI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack), do the following:

- Copy `php4ts.dll` to your systemroot (the directory where you installed windows)
- Make a file association from the command line. Type the following two lines:

  ```
  assoc .php=PHPScript
  ftype PHPScript=c:\php\php.exe %1 %*
  ```

- In the Netscape Enterprise Administration Server create a dummy shellcgi directory and remove it just after (this step creates 5 important lines in obj.conf and allow the web server to handle shellcgi scripts).
- In the Netscape Enterprise Administration Server create a new mime type (Category: type, Content-Type: magnus-internal/shellcgi, File Suffix:php).
- Do it for each web server instance you want php to run

More details about setting up PHP as a CGI executable can be found here: http://benoit.noss.free.fr/php/install-php.html

To Install PHP as NSAPI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack, do the following:

- Copy `php4ts.dll` to your systemroot (the directory where you installed windows)
- Make a file association from the command line. Type the following two lines:

  ```
  assoc .php=PHPScript
  ftype PHPScript=c:\php\php.exe %1 %*
  ```

- In the Netscape Enterprise Administration Server create a new mime type (Category: type, Content-Type: magnus-internal/x-httpd-php, File Suffix:php).
- Stop your web service and edit `obj.conf`. At the end of the Init section, place these two lines (necessarily after mime type init!):

  ```
  Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib=
  Init fn="php4_init" errorString="Failed to initialise PHP!"
  ```

- In The `< Object name="default" >` section, place this line necessarily after all 'ObjectType' and before all 'AddLog' lines:

  ```
  Service fn="php4_execute" type="magnus-internal/x-httpd-php"
  ```

- At the end of the file, create a new object called `x-httpd-php`, by inserting these lines:

  ```
  <Object name="x-httpd-php">
  ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
  Service fn=php4_execute
  </Object>
  ```

- Restart your web service and apply changes
- Do it for each web server instance you want PHP to run

More details about setting up PHP as an NSAPI filter can be found here:
http://benoit.noss.free.fr/php/install-php4.html

Servers-OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd.

OmniHTTPd 2.0b1 and up for Windows

You need to complete the folowing steps to make PHP work with OmniHTTPd. This is a CGI executable setup. SAPI is supported by OmniHTTPd, but some tests showed, that it is not so stable to use PHP as an ISAPI module in OmniHTTPd.

- Step 1: Install OmniHTTPd server.
- Step 2: Right click on the blue OmniHTTPd icon in the system tray and select `Properties`
- Step 3: Click on `Web Server Global Settings`
- Step 4: On the 'External' tab, enter: `virtual = .php | actual = c:\path-to-php-dir\php.exe`, and use the Add button.
- Step 5: On the `Mime` tab, enter: `virtual = wwwserver/stdcgi | actual = .php`, and use the Add button.
- Step 6: Click `OK`

Repeat steps 2 - 6 for each extension you want to associate with PHP.

> **Anmerkung:** Some OmniHTTPd packages come with built in PHP support. You can choose at setup time to do a custom setup, and uncheck the PHP component. We recommend you to use the latest PHP binaries. Some OmniHTTPd servers come with PHP 4 beta distributions, so you should choose not to set up the built in support, but install your own. If the server is already on your machine, use the Replace button in Step 4 and 5 to set the new, correct information.

Servers-Oreilly Website Pro

This section contains notes and hints specific to Oreilly Website Pro.

Oreilly Website Pro 2.5 and up for Windows

This list describes how to set up the PHP CGI binary or the ISAPI module to work with Oreilly Website Pro on Windows.

- Edit the Server Properties and select the tab "Mapping".
- From the List select "Associations" and enter the desired extension (`.php`) and the path to the CGI exe (ex. `c:\php\php.exe`) or the ISAPI DLL file (ex. `c:\php\sapi\php4isapi.dll`).
- Select "Content Types" add the same extension (`.php`) and enter the content type. If you choose the CGI executable file, enter 'wwwserver/shellcgi', if you chosse the ISAPI module, enter 'wwwserver/isapi' (both without quotes).

Servers-Xitami

This section contains notes and hints specific to Xitami.

Xitami for Windows

This list describes how to set up the PHP CGI binary to work with Xitami on Windows.

- Make sure the webserver is running, and point your browser to xitamis admin console (usually `http://127.0.0.1/admin`), and click on Configuration.
- Navigate to the Filters, and put the extension which PHP should parse (i.e. .php) into the field File extensions (.xxx).
- In Filter command or script put the path and name of your php executable i.e. `c:\php\php.exe`.
- Press the 'Save' icon.

Servers-Other web servers

PHP can be built to support a large number of web servers. Please see Server-related options for a full list of server-related configure options. The PHP CGI binaries are compatible with almost all webservers supporting the CGI interface.

Problems?

Read the FAQ

Some problems are more common than others. The most common ones are listed in the PHP FAQ, found at http://www.php.net/FAQ.php

Other problems

If you are still stuck, someone on the PHP installation mailing list may be able to help you. You should check out the archive first, in case someone already answered someone else who had the same problem as you. The archives are available from the support page on http://www.php.net/. To subscribe to the PHP installation mailing list, send an empty mail to php-install-subscribe@lists.php.net (mailto:php-install-subscribe@lists.php.net). The mailing list address is `php-install@lists.php.net`.

If you want to get help on the mailing list, please try to be precise and give the necessary details about your environment (which operating system, what PHP version, what web server, if you are running PHP as CGI or a server module, etc.), and preferably enough code to make others able to reproduce and test your problem.

Bug reports

If you think you have found a bug in PHP, please report it. The PHP developers probably don't know about it, and unless you report it, chances are it won't be fixed. You can report bugs using the bug-tracking system at http://bugs.php.net/.

Read the Bugs-Dos-And-Donts (http://bugs.php.net/bugs-dos-and-donts.php) before submitting any bug reports!

Kapitel 3. Konfiguration

Die Konfigurationsdatei

Die Konfigurationsdatei (`php3.ini` in PHP 3.0.x, und `php.ini` in PHP 4.0) wird geladen, wenn PHP gestartet wird. Wurde PHP als Modul in den Webserver einkompiliert, dann geschieht dies nur, wenn der Server gestartet wird. Ist PHP als CGI-Version konfiguriert worden, geschieht es bei jedem Aufruf.

Wenn Sie PHP in der Modul-Variante benutzen, können Sie die Konfigurationseinstellungen auch mittels der Apache-Konfigurations- datei bzw. mittels .htaccess-Dateien ändern.

Bei PHP 3.0.x existieren Apache-Anweisungen, die mit jeder Konfigurationseinstellung der `php3.ini` korrespondieren, ausgenommen Einstellungen mit dem Prefix "php3_".

Bei PHP 4.0 gibt es nur noch wenige Apache-Anweisungen, die es Ihnen erlauben, die Konfigurationseinstellungen zu ändern.

php_value *`name value`*

Dieses setzt den Wert der spezifizierten Variablen.

php_flag *`name on|off`*

Dieser Schalter wird benutzt, um die Boolean-Konfigurations- Option zu aktivieren.

php_admin_value *`name value`*

Dieser Wert setzt den Wert der spezifischen Variablen. "Admin"-Konfigurationseinstellungen können nur innerhalb der Haupt-Konfigurationsdatei des Apache gesetzt werden, nicht etwa über eine .htaccess Datei.

php_admin_flag *`name on|off`*

Dieser Schalter wird benutzt, um die Boolean-Konfigurations-Option zu aktivieren. ("Admin")

Die Konfigurationseinstellungen können Sie in der Ausgabe der phpinfo() Datei einsehen. Ebenfalls können Sie Zugang zu den individuellen Konfigurationseinstellungen über get_cfg_var() bekommen.

Allgemeine Konfigurationseinstellungen

`asp_tags` boolean

Dieser Schalter aktiviert die Unterstützung von ASP <% %> Tags als Erweiterung zu den üblichen <?php ?> Tags. Dieses beinhaltet auch die Kurzform der Variablenausgabe <%= $value %>. Weitere Informationen finden Sie hier: Escaping from HTML.

Anmerkung: Die Unterstützung für ASP-Tags wurde in Version 3.0.4 hinzugefügt.

`auto_append_file` string

Hier können Sie eine Datei angeben, die automatisch nach der Haupt-Datei aufgerufen wird. Die Datei wird in die aufgerufene Datei eingebettet, als wenn Sie sie mittels der include() Funktion eingebunden hätten, dabei wird ein gesetzter include_path beachtet.

Der Wert none deaktiviert auto-appending.

Anmerkung: Wird das Skript mit der Funktion exit()beendet, wird auto-append *nicht* aktiv.

`auto_prepend_file` string

Hier können Sie eine Datei angeben, die automatisch vor der Hauptdatei aufgerufen wird. Die Datei wird in die aufgerufene Datei eingebettet, als wenn Sie sie mittels der include() Funktion eingebunden hätten, dabei wird ein gesetzter include_path beachtet.

Der Wert none deaktiviert auto-prepending.

`cgi_ext` string

`display_errors` boolean

Dieser Wert muß "on" sein, damit Fehlermeldungen an die Konsole (Prompt oder Browser) gesendet werden können.

`doc_root` string

Hier wird das Stammverzeichnis der PHP-Skripte eingegeben. Üblicherweise handelt es sich hier um das DocumentRoot des Servers (Apache: `htdocs`) Diese Angabe wird nur benutzt, wenn sie einen Wert enthält. Wenn PHP mit safe mode konfiguriert wurde, werden alle PHP-Skripte außerhalb dieses Directorys ignoriert.

`engine` boolean

Diese Option ist in erster Linie nur sinnvoll, wenn PHP als Modul in den Apache einkompiliert wurde. Sie wird von Seiten benutzt, die den PHP-Parser für bestimmte Verzeichnisse oder virtuelle Server aus- bzw. einschalten wollen. Wenn Sie **`engine off`** in den dafür vorgesehenen Blöcken in der `httpd.conf` Datei benutzen, kann PHP aktiviert bzw. deaktiviert werden.

`error_log` string

Hier können Sie die Datei angeben, in der Skript-Fehler protokolliert werden sollen. Wenn Sie statt eines Dateinamens `syslog` eintragen, wird stattdessen das Ereignisprotokoll von WindowsNT genutzt. Auf UNIX Systemen syslog(3) verwendet. Windows 95/98 unterstützen dieses nicht.

`error_reporting` integer

Hier können Sie die Genauigkeit der Fehlermeldungen einstellen. Der eingetragene Wert ist ein Bitwert, und wird als Summe folgender Werte des Error Reporting Levels gebildet:

Tabelle 3-1. Error Reporting Levels

bit value	Aktivierte Protokollierung
1	Normale Fehler

bit value	Aktivierte Protokollierung
2	Normale Warnungen
4	Fehler des Parsers, meistens Syntaxfehler
8	Unkritische Warnungen, die ignoriert werden können. Sinnvoll in der Entwicklungsphase.

Standardwert ist hier 7 (normale Fehler, normale Warnungen, Fehler des Parsers werden angezeigt).

open_basedir string

Beschränkt die Dateien, die von PHP geöffnet werden können, auf Dateien im angegebenen Verzeichnisbaum.

Wenn ein Skript versucht, eine Datei mit z.B. fopen oder gzopen zu öffnen, wird der Ort der Datei überprüft. Wenn sich die Datei außerhalb des spezifizierten Verzeichnisses befindet, wird PHP sie nicht öffnen. Alle symbolischen Links sind hier mit eingeschlossen, so dass es auch nicht möglich ist, dieses Verbot mittels symlink zu umgehen.

Der Wert . gibt an, dass das Verzeichnis, in dem das Skript abgespeichert ist, als Basis-Verzeichnis genutzt wird.

Unter Windows werden Verzeichnisse mit Semikolon getrennt, unter allen anderen Betriebssystemen mit einem Doppelpunkt. Wenn PHP als Modul in den Apache einkompiliert wurde, werden open_basedir paths von "Eltern"-Verzeichnissen nun automatisch vererbt.

> **Anmerkung:** Die Unterstützung für multiple Verzeichnisse wurde in Version 3.0.7 hinzugefügt.

Die Standardeinstellung ist, alle Dateien zum Parsen freizugeben.

gpc_order string

Legt die Auswertungsreihenfolge für GET/POST/COOKIE Variablen fest. Die Standardeinstellung ist GPC. Wenn Sie diesen Wert auf z.B."GP" setzen, ignoriert PHP Cookies, und wird jede GET Methoden Variable mit POST Methoden Variablen des gleichen Namens überschreiben.

ignore_user_abort string

Standardmäßig auf ON. Wenn Sie diese Einstellung auf OFF setzen, werden PHP-Skripte beendet, sobald sie versuchen, eine Ausgabe zu erzeugen, nachdem der Client die Verbindung beendet hat. ignore_user_abort().

include_path string

Hier können Sie ein Verzeichnis angeben, in dem die require(), include() und **fopen_with_path()** Funktionen nach Dateien suchen. Das Format ist ähnlich der PATH Umgebungsvariablen des jeweiligen Systems: Eine Liste von Verzeichnissen, getrennt durch einen Doppelpunkt bei UNIX oder ein Semikolon bei WINDOWS.

Beispiel 3-1. UNIX include_path

```
include_path=.:/home/httpd/php-lib
```

Beispiel 3-2. Windows include_path

```
include_path=".;c:\www\phplib"
```

Der Standardwert für diese Option ist `.` (nur das aktuelle Verzeichnis).

`isapi_ext` string

`log_errors` boolean

Dieser Schalter entscheidet, ob Skript-Fehler im Fehler-Logfile des Servers protokolliert werden sollen. Diese Option ist also Server-spezifisch.

`magic_quotes_gpc` boolean

Dieser Schalter setzt den magic_quotes-Zustand für GPC (GET/POST/COOKIE) -Operationen. Wenn magic_quotes auf ON steht, werden automatisch alle ' (single-quote), " (double quote), \ (backslash) und NUL's mit einem Backslash versehen. Wenn auch magic_quotes_sybase auf ON steht, wird ein single-quote mit einem weiteren single-quote anstatt eines Backslashs versehen.

`magic_quotes_runtime` boolean

Wenn der Schalter *`magic_quotes_runtime`* aktiviert ist, werden Anführungszeichen der meisten Funktionen, welche Daten aus jeglicher Art von externer Quelle, eingeschlossen Datenbanken und Textdateien, zurückgeben, mit einem Backslash versehen. Wenn auch der Schalter *`magic_quotes_sybase`* aktiviert ist, wird ein einfaches Anführungszeichen mit einem einfachen Anführungszeichen anstatt eines Backslashes versehen.

`magic_quotes_sybase` boolean

Wenn der Schalter *`magic_quotes_sybase`* zusätzlich zu *`magic_quotes_gpc`* oder *`magic_quotes_runtime`* aktiviert ist, werden einfache Anführungszeichen mit einfachen Anführungszeichen anstatt eines Backslashs versehen.

`max_execution_time` integer

Dieser Wert gibt die maximale Ausführungslänge eines Skriptes in Sekunden an, bevor es vom Parser abgebrochen wird. Dieses soll Abstürzen des Servers durch schlecht geschriebene oder falsche Skripte vorbeugen. Bei komplexen Datenbankabfragen sollte man jedoch bedenken, dass der Standardwert unter Umständen zu klein sein kann.

`memory_limit` integer

Dieser Wert gibt den maximal erlaubten Speicherplatzverbrauch eines PHP-Skriptes an. Diese Einstellung soll den Server vor Speicherverschwendung durch schlecht programmierte Skripte bewahren.

`nsapi_ext` string

`register_globals` boolean

Dieser Schalter bestimmt, ob die EGPCS-Variablen (Environment, GET, POST, Cookie, Server) als globale Variablen registriert werden oder nicht. Sie sollten dies ausschalten, wenn der globale Bereich Ihrer Skripte nicht durch Benutzerdaten gestört werden soll. Diese Einstellung ist insbesondere dann sinnvoll, wenn sie mit track_vars verwendet wird. In diesem

Fall kÖnnen alle EGPCS-Variablen durch `$HTTP_ENV_VARS`, `$HTTP_GET_VARS`, `$HTTP_POST_VARS`, `$HTTP_COOKIE_VARS` und `$HTTP_SERVER_VARS` angespochen werden.

`short_open_tag` boolean

Gibt an, ob die Kurzform (**`<? ?>`** eines öffnenden PHP-Tags erlaubt ist. Wenn Sie PHP in Kombination mit XML nutzen, müssen Sie diese Option deaktivieren. Wenn Sie die Option deaktiviert haben, müssen Sie die Langform des öffnenden Tags (**`<?php ?>`**)verwenden.

`sql.safe_mode` boolean

`track_errors` boolean

Wenn diese Option aktiviert ist, wird der letzte Fehler immer in der globalen Variablen $php_errormsg abgelegt.

`track_vars` boolean

Wenn dieser Schalter aktiviert ist, werden GET-, POST- und Cookie-Werte in den Umgebungsvariablen-Arrays $HTTP_GET_VARS, $HTTP_POST_VARS und $HTTP_COOKIE_VARS abgelegt.

`upload_tmp_dir` string

Hier wird das temporäre Verzeichnis angegeben, in welchem Dateien gespeichert werden, die mittels file upload auf den Server geladen werden. Dieses Verzeichnis muss schreibbar sein.

`user_dir` string

Wenn Sie PHP für User hosten, dann können persönliche Verzeichnisse der Art ~username angelegt werden. Der Wert user_dir zeigt auf das Stammverzeichnis, unter dem diese Nutzerverzeichnisse liegen. z.B. `public_html`.

`warn_plus_overloading` boolean

Wenn dieser Schalter aktiviert ist, gibt PHP eine Warnung aus, wenn der Plus-Operator (`+`) anstatt des Punkt-Operators (`.`) für die String-Verkettung benutzt wurde.

Mail-Konfigurationsdirektiven

`SMTP` string

Dieser Wert muss nur in der Windows-Umgebung gesetzt werden. Hier wird der DNS-Name oder die IP-Adresse des SMTP-Servers eingetragen, über den Mails mit der Funktion mail() verschickt werden sollen.

`sendmail_from` string

Hier können Sie den Absender der Emails eintragen, wenn Sie von Windows aus Emails verschicken wollen.

`sendmail_path` string

Hier geben Sie den Pfad zum **sendmail** Programm an. Für gewöhnlich ist dies `/usr/sbin/sendmail` oder `/usr/lib/sendmail`. **configure** versucht zwar den Pfad zum

sendmail Programm selbstständig herauszufinden, aber für den Fall, dass dieses nicht funktioniert, können Sie den Pfad hier eintragen.

Auf Systemen, wo Sendmail nicht benutzt wird, sollten Sie diesen Pfad auf den Sendmail-Ersatz -wenn vorhanden- setzen. Zum Beispiel: Qmail (http://www.qmail.org/)-Benutzer können den Pfad normalerweise wie folgt setzen: `/var/qmail/bin/sendmail`.

Safe Mode-Konfigurationsdirektiven

`safe_mode` boolean

Wenn Sie diesen Schalter auf ON setzen, betreiben Sie PHP im sicheren Modus. Mehr dazu erfahren Sie im Kapitel Sicherheit

`safe_mode_exec_dir` string

Wenn Sie PHP im sicheren Modus betreiben, werden die Funktion system() und andere Funktionen, die Systemprogramme ausführen, nicht gestartet, wenn Sie sich nicht in diesem Verzeichnis befinden.

Debugger-Konfigurationsdirektiven

`debugger.host` string

DNS-Name oder IP-Adresse des Hosts, der vom Debugger benutzt wird.

`debugger.port` string

Portnummer, die vom Debugger benutzt wird.

`debugger.enabled` boolean

Hier können Sie den Debugger aktivieren bzw. deaktivieren.

Extension Loading-Direktiven

`enable_dl` boolean

Diese Option ist eigentlich nur sinnvoll, wenn Sie PHP als Apache-Modul benutzen. Dann können Sie das "Einladen" der dynamischen PHP-Extensionen mit der Funktion dl() für einzelne virtuelle Server oder Verzeichnisse aktivieren bzw. deaktivieren.

Der Hauptgrund, das dynamische Laden von Extensionen zu deaktivieren, ist Sicherheit. Wenn die Option aktiviert ist, ist es möglich, Restriktionen durch safe_mode und open_basedir zu umgehen.

Standardmäßig wird das dynamische Laden erlaubt, außer wenn Sie PHP im safe_mode betreiben. Im safe_mode ist es nie möglich, die Funktion dl() zu benutzen.

extension_dir string

Hier geben Sie das Verzeichnis an, in dem PHP dynamisch zu ladende Extensionen findet.

extension string

Hier geben Sie an, welche Extensionen geladen werden sollen, wenn PHP gestartet wird.

MySQL-Konfigurationsdirektiven

mysql.allow_persistent boolean

Erlaubt persistente Verbindungen.

mysql.default_host string

Hier wird der Host (Computername) des Computers eingetragen, auf dem MySQL installiert ist (wird benutzt, falls kein anderer Name angegeben wird).

mysql.default_user string

Hier wird der Name des Standardbenutzers eingegeben (wird benutzt, falls kein anderer Name angegeben wird).

mysql.default_password string

Hier wird das Standardpasswort eingegeben (wird benutzt, falls kein anderes Passwort angegeben wurde).

mysql.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter MySQL-Verbindungen pro Prozess an.

mysql.max_links integer

Hier geben Sie die maximale Anzahl der Verbindungen pro Prozess an (persistente und normale).

mSQL-Konfigurationsdirektiven

msql.allow_persistent boolean

Erlaubt persistente mSQL-Verbindungen.

msql.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter mSQL-Verbindungen pro Prozess an.

msql.max_links integer

Hier geben Sie die maximale Anzahl der Verbindungen pro Prozess an (persistente und normale).

PostgreSQL-Konfigurationsdirektiven

pgsql.allow_persistent boolean

Erlaubt persistente PostgreSQL-Verbindungen.

pgsql.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter PostgreSQL-Verbindungen pro Prozess an.

pgsql.max_links integer

Hier geben Sie die maximale Anzahl der Verbindungen pro Prozess an (persistente und normale).

SESAM Konfigurationsdirektiven

sesam_oml string

Name der BS2000 PLAM-Bibliothek, die die ladbaren SESAM Treibermodule enthält. Diese Direktive ist obligatorisch, anderenfalls wird die SESAM Funktionalität deaktiviert. SESAM functions. Die angegebene BS2000 PLAM-Bibliothek muss durch Setzen von ACCESS=READ,SHARE=YES lesbar gemacht werden, da unter der Apache Server User-Id auf sie zugegriffen wird.

sesam_configfile string

Name der SESAM Applikations-Konfigurationsdatei. Diese Direktive ist obligatorisch, anderenfalls wird die SESAM Funktionalität deaktiviert. Die angegebene BS2000-Datei muss für die Apache Server-Kennung lesbar sein.

Gewöhnlich enthält die Applikations-Konfigurationsdatei Konfigurationsbefehle wie (siehe SESAM Referenzhandbuch):

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

Name der SESAM Message-Katalogdatei. In den meisten Fällen ist die Angabe dieser Datei überflüssig; lediglich wenn die Message-Katalogdatei nicht in der BS2000 Messagedatei-Tabelle eingetragen ist, kann sie explizit mit dieser Direktive gesetzt werden.

Die angegebene BS2000 Message-Katalogdatei muss durch Setzen von ACCESS=READ,SHARE=YES lesbar gemacht werden, da unter der Apache Server User-Id auf sie zugegriffen wird.

Sybase-Konfigurationsdirektiven

sybase.allow_persistent boolean

Erlaubt persistente Sybase-Verbindungen.

sybase.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Sybase-Verbindungen pro Prozess an.

sybase.max_links integer

Hier geben Sie die maximale Anzahl der Verbindungen pro Prozess an (persistente und normale).

Sybase-CT-Konfigurationsdirektiven

sybct.allow_persistent boolean

Erlaubt persistente Sybase-CT Verbindungen. Standardwert ist ON.

sybct.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Sybase-Verbindungen pro Prozess an. Der Standardwert ist -1 (unbegrenzt).

sybct.max_links integer

Hier geben Sie die maximale Anzahl der Verbindungen pro Prozess an (persistente und normale). Standardwert ist -1 (unbegrenzt).

sybct.min_server_severity integer

Server-Meldungen mit höherem oder gleichem Gewicht als sybct.min_server_severity werden als Warnung ausgegeben. Dieser Wert kann auch in einem Skript mittels der sybase_min_server_severity() Funktion gesetzt werden. Standardwert ist 10, wodurch Fehler mit Informations-"Gewicht" oder höher ausgegeben werden.

sybct.min_client_severity integer

Meldungen der Client-Library mit höherem oder gleichem Gewicht als sybct.min_client_severity werden als Warnungen ausgegeben. Dieser Wert kann auch in einem Skript mit der Funktion sybase_min_client_severity() gesetzt werden. Der Standardwert ist hier 10, was eine Ausgabe im Endeffekt deaktiviert.

sybct.login_timeout integer

Hier können Sie die maximale Zeit in Sekunden angeben, die gewartet wird, bis eine Verbindung erfolgreich aufgebaut wurde (also bevor eine Fehlermeldung erscheint). Beachten Sie, dass wenn die max_execution_time für einen Verbindungsversuch vorbei ist, Ihr Skript beendet wird, bevor es eine Fehlermeldung (oder andere Aktion) ausgeben kann. Der Standardwert ist hier 1 Minute.

sybct.timeout integer

Hier können Sie die maximale Zeit in Sekunden angeben, die auf eine select_db oder andere DB-Query gewartet wird, bis eine Fehlermeldung erscheint. Beachten Sie, dass wenn die

max_execution_time für eine DB-Query vorbei ist, Ihr Skript beendet wird, bevor es eine Fehlermeldung (oder andere Aktion) ausgeben kann. Der Standardwert ist hier unbegrenzt.

sybct.hostname string

Hier können Sie den Host angeben, von dem die Verbindung ausgehen soll. Sie können diesen Wert mit sp_who anzeigen. Standardwert ist none.

Informix-Konfigurationsdirektiven

ifx.allow_persistent boolean

Erlaubt persistente Informix-Verbindungen.

ifx.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Informix-Verbindungen pro Prozess an.

ifx.max_links integer

Hier geben Sie die maximale Anzahl der Verbindungen pro Prozess an (persistente und normale).

ifx.default_host string

Hier geben Sie den Standardhost für eine Verbindung ein, der benutzt wird, wenn kein anderer Host in ifx_connect() oder ifx_pconnect() angegeben wurde.

ifx.default_user string

Hier geben Sie den Standardbenutzer für eine Verbindung ein, der benutzt wird, wenn kein anderer Benutzer in ifx_connect() oder ifx_pconnect() angegeben wurde.

ifx.default_password string

Hier geben Sie das Standardpasswort für eine Verbindung ein, das benutzt wird, wenn kein anderes Passwort in ifx_connect() oder ifx_pconnect() angegeben wurde.

ifx.blobinfile boolean

Setzen Sie diesen Wert auf `TRUE`, wenn Sie BLOB-Felder in einer Datei zurückgegeben haben wollen, auf `FALSE`, wenn Sie sie im Speicher haben wollen. Diesen Eintrag können Sie mit ifx_blobinfile_mode() während der Ausführung überschreiben.

ifx.textasvarchar boolean

Setzen Sie diesen Wert auf `TRUE`, wenn Sie TEXT-Felder als normale Strings in SELECT-Anweisungen zurückgegeben haben wollen, auf `FALSE`, wenn Sie sie BLOB-ID Parameter benutzen wollen. Diesen Eintrag können Sie mit ifx_textasvarchar() während der Ausführung überschreiben.

ifx.byteasvarchar boolean

Setzen Sie diesen Wert auf `TRUE`, wenn Sie BYTE-Felder in SELECT-Abfragen als normale Strings zurückgeliefert haben wollen, auf `FALSE`, wenn Sie BLOB-ID Parameter benutzen wollen. Diesen Eintrag können Sie mit ifx_textasvarchar() während der Ausführung überschreiben.

ifx.charasvarchar boolean

Setzen Sie diesen Wert auf TRUE, wenn Sie nachfolgende Spaces von CHAR-Feldern bei der Abfrage abschneiden wollen.

ifx.nullformat boolean

Setzen Sie diesen Wert auf TRUE, wenn Sie NULL-Felder als String "NULL", bzw. auf FALSE, wenn Sie diese als leeren String "" zurückgeliefert haben wollen. Sie können diesen Wert während der Ausführung mit der Funktion ifx_nullformat() überschreiben.

BC Math-Konfigurationsdirektiven

bcmath.scale integer

Anzahl der Nachkommastellen der bcmath-Funktionen.

Browser Capability-Konfigurationsdirektiven

browscap string

Name der browser_capabilities-Datei. Mehr hierüber erfahren Sie unter get_browser().

Unified ODBC-Konfigurationsdirektiven

uodbc.default_db string

ODBC Standarddatenquelle, die benutzt wird, wenn keine andere in odbc_connect() oder odbc_pconnect() angegeben wurde.

uodbc.default_user string

ODBC Standard-Benutzername, der benutzt wird, wenn kein anderer Benutzer in odbc_connect() oder odbc_pconnect() definiert wurde.

uodbc.default_pw string

ODBC Standardpasswort, das benutzt wird, wenn kein anderes Passwort in odbc_connect() oder odbc_pconnect() angegeben wurde.

uodbc.allow_persistent boolean

Erlaubt persistente ODBC-Verbindungen.

uodbc.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter ODBC-Verbindungen pro Prozess an.

uodbc.max_links integer

Hier geben Sie die maximal erlaubte Anzahl der ODBC-Verbindungen pro Prozess an (persistente und normale).

Kapitel 4. Sicherheit

PHP ist eine mächtige Sprache, und der Interpreter, der in einen Webserver als Modul oder als separate CGI-Version eingebunden werden kann, kann auf Dateien zugreifen, Befehle ausführen und Netzwerkverbindungen zu einem Server herstellen. Diese Eigenschaften können einen Webserver unsicher machen, wenn man es bei den Voreinstellungen belässt. PHP wurde besonders dafür entwickelt, um eine sicherere Sprache als Perl oder C für die Erstellung von CGI-Programmen bereitzustellen. Mit der richtigen Wahl der Einstellungen beim Kompilieren und zur Laufzeit bietet PHP genau die Kombination aus Freiheit und Sicherheit, die gerade benötigt wird.

Da es sehr viele verschiedene Möglichkeiten gibt, PHP zu nutzen, gibt es viele Konfigurationseinstellungen, die das Verhalten von PHP beeinflussen. Eine große Auswahl an Einstellungen garantiert, dass man PHP für viele Zwecke einsetzen kann. Allerdings bedeutet das auch, dass es Kombinationen gibt, die eine Installation mit nur ungenügender Sicherheit zur Folge haben.

Die Flexibilität der Konfiguration konkurriert mit der Flexibilität in der Programmierung. Mit PHP können komplette Server Applikationen mit allen Möglichkeiten eines Shell Benutzers erstellt werden, oder auch nur einfache Server Side Includes mit einem minimalen Risiko in einer streng kontrollierten Umgebung. Wie die Umgebung erstellt wird, und wie sicher diese ist, ist zu einem großen Teil die Sache des PHP Entwicklers.

Dieses Kapitel beginnt mit einigen generellen Ratschlägen zur Sicherheit, erklärt die verschiedenen Kombinationen der Konfigurationseinstellungen und unter welchen Gegebenheiten sie sicher genutzt werden können, und beschreibt verschiedene Überlegungen zur Programmierung für verschiedene Sicherheitsstufen.

Allgemeine Überlegungen

Ein komplett sicheres System ist praktisch ein Ding der Unmöglichkeit, weshalb ein unter Sicherheitsprofis oft genutzter Ansatz ist, einen Mittelweg zwischen Risiko und Verwendbarkeit zu finden. Wenn jede von einem Benutzer übermittelte Variable zwei Formen von biometrischer Prüfung (wie z.B. ein Scan der Netzhaut und ein Fingerabdruck) verlangen würde, wäre eine extrem hohe Ebene der Verantwortlichkeit erreicht. Ein sehr komplexes Formular auszufüllen würde auch eine halbe Stunde in Anspruch nehmen, die Benutzer dazu ermuntern könnte, Wege zur Umgehung der Sicherheitsmaßnahmen zu suchen.

Die beste Sicherheit ist oft unaufdringlich genug um den Anforderungen zu entsprechen, ohne den Benutzer an seiner Arbeit zu hindern oder den Code-Autor mit übertriebener Komplexität zu überlasten. Tatsächlich sind einige Sicherheitsangriffe nur die Folge von allzu strengen Sicherheitsmaßnahmen, was mit der Zeit nur zu deren Unterminierung führt.

Eine Phrase die es wert ist, sich an sie zu erinnern: Ein System ist nur so gut wie das schwächste Glied in der Kette. Wenn alle Transaktionen mittels Zeit, Ort, Transaktionstyp, etc. streng mitprotokolliert werden, der Benutzer aber nur mittels einem einzigen Cookie verifiziert wird, lässt die Zuverlässigkeit für die Bindung des Benutzers an das Transaktions-Log bedrohlich nach.

Denken Sie während der Tests daran, dass Sie selbst für die einfachsten Seiten nicht alle Möglichkeiten testen können. Der von Ihnen vielleicht erwartete Input wird zu dem eines verstimmten Mitarbeiters oder eines Crackers der Monate Zeit hat, oder einer Katze, die über die Tastatur läuft in keinerlei Zusammenhang stehen. Deshalb betrachten Sie Ihren Code am Besten aus der logischen Perspektive um zu erkennen, wo unerwartete Daten eingebracht werden können und fragen sich dann, wie diese modifiziert, reduziert, oder weiter ausgeführt werden.

Das Internet ist voll von Leuten die versuchen, sich durch Entschlüsseln/zerstören Ihres Codes, den Zusammenbruch Ihres Systems, Einsetzen von unangebrachten Inhalten, und anderen, Ihren Tag interessant gestaltenden Maßnahmen, einen Namen zu machen. Es ist egal, ob Sie eine kleine oder

große Site haben, Sie sind einfach ein Ziel wenn Sie online sind oder wenn Sie einen Server haben, zu dem man eine Verbindung aufbauen kann. Viele Cracker-Programme erkennen nicht die Größe, sondern durchsieben einfach gewaltige IP Blöcke im Netz, um Opfer zu finden. Versuchen Sie, keines zu werden.

CGI-Version

Mögliche Angriffe

PHP als CGI zu nutzen, ist eine Möglichkeit für Installationen, bei denen aus irgendwelchen Gründen kein Modul in die Serversoftware eingebunden werden soll (wie beim Apache) oder für Systeme, bei denen verschiedene CGI-Wrapper genutzt werden sollen, um sichere chroot- und setuid-Umgebungen für Skripte zu schaffen. Bei dieser Konfiguration wird das ausführbare PHP-Binary üblicherweise im cgi-bin Verzeichnis des Webservers installiert. CERT advisory CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) spricht sich gegen die Platzierung von Interpretern im cgi-bin Verzeichnis aus. Obwohl das PHP-Binary als eigenständiger Interpreter verwendet werden kann, wurde PHP so entwickelt, um den durch diese Konfiguration möglich werdenden Angriffe vorzubeugen:

- Zugriff auf Systemdateien: `http://my.host/cgi-bin/php?/etc/passwd`

  Die auf ein Fragezeichen (?) folgende Abfrageinformation in einer URL wird durch das CGI-Interface als Kommandozeilenargument an den Interpreter weitergereicht. In der Kommandozeile wird üblicherweise die im ersten Argument angegebene Datei von Interpretern geöffnet und ausgeführt.

  Beim Aufruf als CGI-Binary verweigert PHP die Interpretation der Kommandozeilenargumente.

- Zugriff auf beliebige Web-Dokumente auf dem Server: `http://my.host/cgi-bin/php/secret/doc.html`

  Der Teil der URL-Pfadinformation nach dem Namen der PHP Binärdatei, `/secret/doc.html` wird im allgemeinen benutzt, um den Namen der Datei zu übergeben, die durch das CGI-Programm geöffnet und interpretiert werden soll. Normalerweise werden einige Einträge in der Konfigurationsdatei des Webservers benutzt (Apache: Action), um Aufrufe von Dokumenten wie `http://my.host/secret/script.php3` an den PHP-Interpreter umzuleiten. Bei dieser Konfiguration überprüft der Webserver zuerst die Zugriffsrechte im Verzeichnis `/secret` und erstellt anschließend den umgeleiteten Aufruf `http://my.host/cgi-bin/php/secret/script.php`. Unglücklicherweise wird, wenn der Aufruf bereits in dieser Form geschieht, vom Webserver keine Zugriffsüberprüfung der Datei `/secret/script.php`, sondern lediglich der Datei `/cgi-bin/php` vorgenommen. So ist jeder Benutzer, der auf `/cgi-bin/php` zugreifen darf in der Lage, sich zu jedem geschützten Dokument auf dem Webserver Zugriff zu verschaffen.

  Bei PHP können beim Kompilieren die Konfigurationsoption --enable-force-cgi-redirect und zur Laufzeit die Konfigurationsdirektiven doc_root und user_dir benutzt werden, um diesen Angriff zu verhindern, falls der Verzeichnisbaum des Servers Verzeichnisse mit Zugriffsbeschränkungen beinhaltet. Ausführliche Informationen über die verschiedenen Kombinationen siehe weiter unten.

Fall 1: Nur öffentliche Dateien vorhanden

Wenn der Server keine Inhalte hat, die durch Passwort oder IP-basierte Zugriffskontrolle geschützt sind, werden diese Konfigurationsoptionen nicht benötigt. Wenn der Webserver keine Redirects erlaubt oder keine Möglichkeit hat, dem PHP-Binary mitzuteilen dass es sich um eine sicher umgeleitete Anfrage handelt, kann die Option --enable-force-cgi-redirect im configure-Script angegeben werden. Nichtsdestotrotz müssen Sie sicherstellen, dass Ihre PHP-Skripte nicht auf die eine oder andere Art des Aufrufs angewiesen sind, weder direkt durch `http://my.host/cgi-bin/php/dir/script.php` noch durch einen Redirect `http://my.host/dir/script.php`.

Beim Apache kann der Redirect durch den Gebrauch von AddHandler und Action konfiguriert werden (siehe unten).

Fall 2: --enable-force-cgi-redirect benutzen

Diese beim Kompilieren verwendete Option verhindert grundsätzlich den Aufruf von PHP mit einer URL wie `http://my.host/cgi-bin/php/secretdir/script.php`. Stattdessen parst PHP in diesem Modus nur dann, wenn der Aufruf durch einen korrekten Redirect des Webservers erfolgte.

Normalerweise wird der Redirect in der Apache-Konfiguration mit den folgenden Einträgen festgelegt:

```
Action php-script /cgi-bin/php
AddHandler php-script .php
```

Diese Option wurde nur mit dem Apache Webserver getestet und ist abhängig davon, wie Apache die nicht standardmäßige CGI-Umgebungsvariable REDIRECT_STATUS bei Redirect-Anfragen setzt. Sollte Ihr Webserver keine Möglichkeit unterstützen, zu übermitteln, ob es sich um einen direkte Aufruf oder einen Redirect handelt, können Sie diese Option nicht verwenden und müssen einen der anderen hier beschriebenen Wege gehen, die CGI-Version zu nutzen.

Fall 3: doc_root oder user_dir festlegen

Aktiven Inhalt, wie beispielsweise Skripts und ausführbare Dateien, in den Dokumentverzeichnissen des Webservers abzulegen, wird manchmal als unsichere Methode angesehen. Wenn, beispielsweise aufgrund von Konfigurationsfehlern, die Skripte nicht ausgeführt, sondern als reguläre HTML-Dokumente angezeigt werden kann dies ein Durchsickern von geistigem Eigentum und sicherheitsrelevanter Informationen (Passwörter!) zur Folge haben. Deshalb ziehen es viele Systemadministratoren vor, eine zweite Verzeichnisstruktur für Skripte einzurichten, auf die nur durch das PHP-CGI zugegriffen werden kann. Diese werden dann stets interpretiert und nicht angezeigt.

Auch wenn die Methode zum sichergestellten Verhindern einer Umleitung von Anfragen (wie im vorangegangenen Kapitel beschrieben) nicht verfügbar ist, ist es notwendig, ein doc_root für Skripte zusätzlich zum Web-Dokumentenverzeichnis einzurichten.

Sie können das PHP-Skriptverzeichnis durch die Direktive doc_root in der Konfigurationsdatei festlegen, oder Sie setzen die Umgebungsvariable PHP_DOCUMENT_ROOT. Wenn sie gesetzt ist, wird die CGI-Version von PHP den Namen der zu öffnenden Datei stets aus *`doc_root`* und der

Pfadinformation der Anfrage zusammensetzen, sodass man sicher sein kann, dass außerhalb dieses Verzeichnisses keine Skripte ausgeführt werden (außer *user_dir*, siehe unten).

Eine weitere hier nützliche Option ist user_dir. Wenn das *user_dir* nicht gesetzt ist, hat nur *doc_root* Einfluss auf die zu öffnende Datei. Der Aufruf einer URL wie `http://my.host/~user/doc.php` hat nicht zum Ergebnis, dass eine Datei im Home-Verzeichnis des Benutzers geöffnet wird, sondern eine Datei namens `~user/doc.php` unterhalb des doc_root (Ja, ein Verzeichnisname, der mit einer Tilde anfängt [~]).

Ist das user_dir beispielsweise auf `public_php` gesetzt, wird eine Anfrage wie `http://my.host/~user/doc.php` eine Datei namens `doc.php` im Verzeichnis `public_php` im Heimatverzeichnis des Benutzers öffnen. Wenn das Heimatverzeichnis des Benutzers `/home/user` ist, so ist die ausgeführte Datei `/home/user/public_php/doc.php`.

Die *user_dir*-Expansion erfolgt ohne Berücksichtigung der *doc_root* Einstellung. So können Zugriffe auf die Dokumenten- und Benutzerverzeichnisse separat gesteuert werden.

Fall 4: PHP-Parser außerhalb des Webverzeichnisbaums

Eine sehr sichere Sache ist es, das PHP-Parser-Binary irgendwo außerhalb des Webverzeichnisbaums zu platzieren, beispielsweise in `/usr/local/bin`. Der einzige Nachteil dieses Verfahrens ist, dass eine Zeile ähnlich der folgenden:

```
#!/usr/local/bin/php
```

als erste Zeile in jeder Datei, die PHP-Tags enthält, stehen muss. Außerdem muss die Datei ausführbar sein. Ansonsten ist sie genauso zu behandeln wie ein beliebiges CGI-Script in Perl oder sh oder anderen gebräuchlichen Scriptsprachen, die den `#!` shell-escape-Mechanismus nutzen, um sich selbst aufzurufen.

Damit PHP bei dieser Konfiguration die PATH_INFO- und PATH_TRANSLATED-Informationen korrekt auswertet, sollte der PHP-Parser mit der Option --enable-discard-path kompiliert werden.

Apache-Modul

Wenn PHP als Apache-Modul eingesetzt wird, übernimmt es die Benutzerrechte des Apache (üblicherweise user "nobody"). Das hat verschiedene Auswirkungen auf Sicherheit und Authentifizierung, z.B.: Wenn Sie PHP zum Zugriff auf eine Datenbank benutzen, es sei denn diese Datenbank hat eine integrierte Zugriffskontrolle, müssen Sie dem Benutzer "nobody" Zugriffsrechte auf die Datenbank erteilen. Das heißt, dass ein böswilliges Skript auch Benutzerkennung und Passwort auf die Datenbank zugreifen, und sie verändern könnte. Es ist durchaus möglich, dass ein Web-Spider über die Webseite eines Datenbankadministrators stolpert, und alle Ihre Datenbanken löscht. Sie können sich dagegen mit Apache Authentifizierung schützen, oder ein eigenes Zugangsmodell unter Verwendung von LDAP, .htaccess Dateien, etc. entwerfen, und diesen Code als Teil Ihrer PHP Skripte einbinden.

Es wurde festgestellt, dass wenn einmal die Sicherheitsmaßnahmen so weit eingerichtet sind dass dem PHP User (in diesem Fall ein Apache User) nur mehr ein geringes Risiko bleibt, PHP daran gehindert wird, virenverseuchte Dateien in das Benutzerverzeichnis zu schreiben. Oder vielleicht

wurde es auch daran gehindert, auf Datenbanken zuzugreifen oder diese gar zu verändern. In gleicher Weise wird auch davor abgehalten, "gute" oder "bösartige" Dateien zu schreiben, oder "gute" bzw. "bösartige" Datenbanktransaktionen durchzuführen.

Ein häufig gemachter Fehler in Punkto Sicherheit ist Apache Root-Rechte zu erteilen, oder die Möglichkeiten von Apache in einer anderen Weise auszuweiten.

Die Ausweitung der Benutzerrechte für Apache auf root ist sehr gefährlich, und kann dem gesamten System schaden, denn mit sudo, chroot, oder anderwärtig als root zu arbeiten sollte niemand anders als den Sicherheitsprofis überlassen werden.

Es gibt auch ein paar einfachere Lösungen. Mit open_basedir() können Sie kontrollieren, welche Verzeichnisse PHP benutzen darf oder nicht. Sie können auch einen Bereich nur für Apache einrichten, um alle webbasierten Aktivitäten auf nicht-Benutzer- bzw. nicht-System-Dateien einzuschränken.

Dateisystem - Sicherheit

PHP ist von den in den meisten Serversystemen implementierten Sicherheitseinstellungen hinsichtlich der Berechtigungen auf Datei- und Verzeichnisebene abhängig. Dies verleiht Ihnen Kontrolle darüber, welche Dateien in dem Dateisystem gelesen werden dürfen. Vorsicht ist bei weltweit lesbaren Dateien geboten um sicherzustellen, dass diese sicher von allen Usern mit Zugriff auf dieses Dateisystem (nur) gelesen werden können.

Da PHP entwickelt wurde um Zugriffe auf das Dateisystem auf Benutzebene zu erlauben, ist es natürlich auch möglich ein PHP Skript zu schreiben das Ihnen erlaubt Systemdateien wie /etc/password zu lesen, Ethernetverbindungen zu modifizieren, enorme Druckaufträge zu senden, etc. Dies hat offensichtliche Implikationen, indem Sie sicherstellen müssen, dass alle von Ihnen zu lesenden bzw. zu schreibenden Dateien auch die richtigen sind.

Stellen Sie sich folgendes Skript vor, in dem ein User zum Ausdruck bringt, dass gerne eine Datei in seinem Heimatverzeichnis löschen möchte. Dies geht von einer Situation aus, in der ein PHP Web-Interface regelmäßig zum Dateimanagement verwendet wird, und der Apache User ist auch berechtigt, in seinem Heimatverzeichnis Dateien zu löschen.

Beispiel 4-1. Schlechte Variablenprüfung führt zu....

```
<?php
// Löschen einer Datei aus dem Heimatverzeichnis des Users
$username = $HTTP_POST_VARS['user_submitted_name'];
$homedir = "/home/$username";
$file_to_delete = "$userfile";
unlink ($homedir/$userfile);
echo "$file_to_delete wurde gelöscht!";
?>
```

Da der Benutzername über ein User-Formular zu posten ist, kann jeder einen Benutzer- und Dateinamen von jemand anderen übertragen, und Dateien löschen. In diesem Fall empfiehlt es sich, eine andere Form der Authentifizierung zu verwenden. Stellen Sie sich vor was passieren würde, wenn die übertragenen Variablen "../etc/" und "passwd" beinhalten würden. Der Code würde dann effektiv lesen:

Beispiel 4-2. ... Ein Angriff auf das Dateisystem

```
<?php
// löscht eine Datei irgendwo auf der Festplatte, wo der
// Benutzer die nötigen Rechte besitzt. Wenn PHP root hat:
$username = "../etc/";
$homedir = "/home/../etc/";
$file_to_delete = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd wurde gelöscht!";
?>
```

Es gibt zwei wichtige Kriterien die Sie beachten sollten, um diese Dinge zu vermeiden:

- Erteilen Sie dem PHP Web-user (Binärdatei) nur eingeschränkte Rechte.
- Prüfen Sie alle übertragenen Variablen.

Hier ist ein verbessertes Skript:

Beispiel 4-3. Etwas sicherere Prüfung des Dateinamens

```
<?php
// löscht eine Datei von der Festplatte, auf die
// der PHP user Zugriff hat.
$username = $HTTP_SERVER_VARS['REMOTE_USER']; // verwendet eine
                                              // Authentifizierungsmethode
$homedir = "/home/$username";

$file_to_delete = basename("$userfile"); // den Pfad entfernen
unlink ($homedir/$file_to_delete);

$fp = fopen("/home/logging/filedelete.log","+a"); //logge die Löschung
$logstring = "$username $homedir $file_to_delete";
fputs ($fp, $logstring);
fclose($fp);

echo "$file_to_delete wurde gelöscht!";
?>
```

Auch dies nicht völlig makellos. Wenn Ihr Authentifizierungssystem Benutzern erlauben sollte, deren eigene Logins zu kreieren, und ein Benutzer wählt den Login "../etc", ist das System wieder aufgedeckt. Aus diesem Grund ziehen Sie es vielleicht vor, einen besseren Check zu schreiben:

Beispiel 4-4. Sicherere Dateinamensprüfung

```
<?php
$username = $HTTP_SERVER_VARS['REMOTE_USER']; // verwendet eine
                                              // Authentifizierungsmethode
$homedir = "/home/$username";

if (!ereg('^[^./][^/]*$', $userfile))
    die('bad filename'); // "DIE", gehen Sie nicht weiter

if (!ereg('^[^./][^/]*$', $username))
     die('bad username'); // "DIE", gehen Sie nicht weiter
//etc...
?>
```

Abhängig vom Betriebssystem gibt es eine große Anzahl Dateien mit der Sie sich befassen sollten, inklusive Einträge für Geräte (/dev/ oder com1), Konfigurationsdateien (/etc/ Dateien und die .ini Dateien), gut bekannte Verzeichnisse (/home/, My Documents), etc. Aus diesem Grund ist es gewöhnlich einfacher eine Vorgangsweise einzuführen, bei der außer den von Ihnen explizit erlaubten Dingen alles verboten ist.

Datenbank - Sicherheit

Heutzutage sind Datenbanken die Hauptkomponenten jeder Webbasierten Applikation, aufgrund welcher Websites verschiedene dynamische Inhalte anbieten können. Nachdem heikle oder geheime Informationen in solch einer Datenbank gespeichert werden können, sollten Sie deren Schutz ernsthaft bedenken.

Um Informationen zu bekommen oder zu speichern, müssen Sie eine legitime Abfrage senden, das Ergebnis holen, und die Verbindung schließen. Heutzutage ist die allgemein verwendete Abfragesprache für solche Interaktionen die Structured Query Language (SQL). Sehen Sie, wie sich ein Angreifer an einer SQL Abfrage zu schaffen machen kann.

Sie werden merken, dass PHP Ihre Datenbank alleine nicht schützen kann. Die folgenden Abschnitte sind eine Einführung in die Grundlagen, wie man innerhalb von PHP Skripten auf Datenbanken zugreift und diese manipuliert.

Denken Sie an diese einfache Regel: tief gestaffelte Verteidigung. Je mehr Platz Sie den Maßnahmen zum Schutz Ihrer Datenbank geben, desto geringer ist die Wahrscheinlichkeit, dass ein Angreifer Erfolg hat, und gespeicherte Geheiminformationen aufdeckt oder missbraucht. Gutes Design des Datenbankschemas, und die Applikation wird mit Ihren größten Befürchtungen fertig.

Datenbanken designen

Der Erste Schritt liegt immer im Erstellen der Datenbank, außer Sie wollen eine bereits existierende Dritter verwenden. Ist eine Datenbank erstellt, ist sie einem Eigentümer zugewiesen, welcher das Kommando zum Erstellen ausgeführt hat. Gewöhnlich kann nur der Eigentümer (oder ein Superuser) alles mit den Objekten in dieser Datenbank machen, und um anderen Benutzern die Verwendung zu erlauben, müssen Rechte vergeben werden.

Applikationen sollten sich mit der Datenbank nie als deren Eigentümer oder als ein Superuser verbinden, da diese Benutzer jede gewollte Abfrage ausführen können, um z.B. das Schema zu modifizieren (z.B. Tabellen löschen) oder den gesamten Inhalt löschen.

Sie können verschiedene Datenbanknutzer mit sehr limitierten Rechten auf Datenbankobjekte für jeden Aspekt Ihrer Applikation anlegen. Nur die wirklich benötigten Rechte sollten gewährt werden, und vermeiden Sie, dass der gleiche Benutzer in verschiedenen Anwendungsfällen mit der Datenbank interagieren kann. Das heißt, dass Eindringlinge, welche unter Verwendung einer dieser Referenzen Zugriff auf Ihre Datenbank erlangt haben, nur so viele Änderungen durchführen können, wie es Ihre Applikation kann.

Implementieren Sie nicht alle Geschäftslogik in die Webapplikation (z.B. Ihr Skript), sondern tun Sie das im Datenbankschema unter Verwendung von Sichten, Triggern, oder Regeln. Wenn sich das System entwickelt, werden neu zu öffnende Ports zu der Datenbank vorgesehen, und Sie müssen die Logik in jedem Datenbank-Client neu implementieren. Überdies können Trigger verwendet werden, um transparent und automatisch mit Feldern umzugehen, welche beim debuggen Ihrer Applikation oder beim Zurückverfolgen von Transaktionen oft einen Einblick gewähren.

Zur Datenbank verbinden

Vielleicht wollen Sie die Verbindungen über SSL herstellen, um die Client/Server Kommunikation für eine erhöhte Sicherheit zu verschlüsseln, oder aber auch ssh verwenden, um die Netzwerkverbindung zwischen den Clients und dem Datenbankserver zu verschlüsseln. Ist eines davon realisiert, wird ein Monitoring Ihres Verkehrs und das Erlangen von Informationen zu harter Arbeit.

Verschlüsseltes Speichermodell

SSL/SSH schützt zwar die gerade auf dem Weg befindlichen Daten vom Client zum Server, jedoch nicht die dauernd in einer Datenbank gespeicherten Daten. SSL ist ein "auf-der-Leitung" Protokoll.

Hat ein Angreifer direkten Zugriff auf Ihre Datenbank (den Webserver umgehend), können die gespeicherten heiklen Daten aufgedeckt oder zweckentfremdet werden, außer wenn die Information von der Datenbank selbst geschützt ist. Die Daten zu verschlüsseln ist ein guter Weg, diese Gefahr zu mildern, doch bieten nur wenige Datenbanken diese Art der Verschlüsselung von Daten.

Der einfachste Weg, dieses Problem zu umgehen ist, erst einmal Ihr eigenes Verschlüsselungspaket zu erstellen, und dieses dann in Ihren PHP Skripten zu nutzen. PHP kann Ihnen in diesem Fall mit seinen verschiedenen Erweiterungen helfen, wie z.B. Mcrypt and Mhash, welche eine große Auswahl an Verschlüsselungsalgorhythmen abdecken. Das Skript verschlüsselt die Daten vor dem Speichern, und entschlüsselt diese wieder beim Erhalt. Siehe die Verweise für weitere Beispiele, wie Verschlüsselung arbeitet.

Im Fall von wirklich versteckten Daten, wenn deren unverfälschte Repräsentation nicht nötig ist (z.B. keine Anzeige), ist hashing ebenfalls überlegenswert. Das bekannte Beispiel für das Hashing ist das Speichern des MD5 hash eines Passwortes in einer Datenbank, anstatt des Passwortes selbst. Siehe auch crypt() und md5().

Beispiel 4-5. Verwenden eines hashed Passwortfeldes

```
// Speichern des Passwort hash
$query  = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",
```

```
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

// Afragen, ob der User das richtige Passwort übermittelt hat
$query = sprintf("SELECT 1 FROM users WHERE name='%s' AND pwd='%s';",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

if (pg_numrows($result) > 0) {
    echo "Welcome, $username!";
}
else {
    echo "Authentication failed for $username.";
}
```

SQL Injection

Viele Entwickler sind sich nicht bewusst, wie man sich an SQL Abfragen zu schaffen machen kann und nehmen an, dass eine SQL Abfrage ein vertrauenswürdiges Kommando ist. Das heißt, dass SQL Abfragen Zugriffskontrollen hinters Licht führen, und dadurch Standard Authentifizierungs- und Authorisationschecks umgehen können, und manchmal können SQL Abfragen sogar Zugriff zu Kommandos auf Betriebssystemebene erlauben.

Direkt SQL Command Injection ist eine Technik, wo ein Angreifer SQL Kommandos erstellt oder existierende verändert, um versteckte Daten sichtbar zu machen, wertvolle Daten zu überschreiben, oder sogar gefährliche Kommandos auf Systemebene des Datenbank-Hosts auszuführen. Dies wird durch die Applikation erreicht, welche den Input des Benutzers mit statischen Parametern kombiniert, um eine SQL Abfrage zu erstellen. Die folgenden Beispiele basieren - leider - auf wahren Begebenheiten.

Dank dem Mangel an Input Validierungen, und dem Verbinden zum Datenbankserver als ein Superuser oder jemand der Benutzer anlegen kann, kann ein Angreifer einen Superuser in Ihrer Datenbank anlegen.

Beispiel 4-6. Die Ergebnisliste in mehrere Seiten aufsplitten ... und Superuser anlegen (PostgreSQL and MySQL)

```
$offset = argv[0]; // Vorsicht, keine Validierung des Input !
$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";
// mit PostgreSQL
$result = pg_exec($conn, $query);
// mit MySQL
$result = mysql_query($query);
```

Normale Benutzer klicken auf die 'nächste' bzw. 'vorige' Links, wo `$offset` in der URL enthalten ist. Das Skript erwartet, dass die ankommende `$offset` einen Dezimalwert enthält. Ganz gleich, jemand versucht einzubrechen, indem er das folgende in einer urlencode()'d Form an die URL anhängt

```
// Im Fall von PostgreSQL
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select 'crack', usesysid, 't','t','crack'
    from pg_shadow where usename='postgres';
--

// Im Fall von MySQL
0;
UPDATE user SET Password=PASSWORD('crack') WHERE user='root';
FLUSH PRIVILEGES;
```

Wenn es passiert ist, würde ihm das Skript einen Zugriff als Superuser präsentieren. Beachten Sie, dass `0;` ein gültiges Offset zur ursprünglichen Abfrage liefert, und sie beendet.

> **Anmerkung:** Es ist eine übliche Technik, den SQL Parser mittels dem Kommentarzeichen in SQL `--` zu zwingen, den Rest der vom Entwickler geschriebenen Abfrage zu ignorieren.

Ein gangbarer Weg um Passwörter zu finden ist, Ihre Seiten mit den Suchergebnissen hinters Licht zu führen. Der Angreifer braucht nur zu probieren, ob irgendeine übertragene Variable, die in dem SQL Statement verwendet wird, nicht richtig gehandhabt wird. Diese Filter können gewöhnlich in einer vorausgehenden Form gesetzt werden, indem `WHERE, ORDER BY, LIMIT` und `OFFSET` Klauseln in `SELECT` Statements umgebaut werden. Wenn Ihre Datenbank das `UNION` Konstrukt unterstützt, kann der Angreifer versuchen, eine komplette Abfrage an das Original anzuhängen, um Paßwörter aus einer willkürlichen Tabelle aufzulisten. Die Verwendung von verschlüsselten Passwortfeldern wird ausdrücklich empfohlen.

Beispiel 4-7. Artikel auflisten ... und ein paar Passwörter (irgendein Datenbankserver)

```
$query  = "SELECT id, name, inserted, size FROM products
                  WHERE size = '$size'
                  ORDER BY $order LIMIT $limit, $offset;";
$result = odbc_exec($conn, $query);
```

Der statische Teil der Abfrage kann mit einem anderen `SELECT` Statement kombiniert werden, welches alle Passwörter preisgibt

```
'
union select '1', concat(uname||'-'||passwd) as name, '1971-01-01', '0' from usertable;
--
```

Wenn diese Abfrage (mit dem `'` und `--`) einer der in `$query` verwendeten Variablen zugewiesen würde, wäre das "Abfragebiest" erwacht.

SQL UPDATEs sind ebenfalls ein Anlass, Ihre Datenbank anzugreifen. Diese Abfragen sind auch durch das Ändern und Anhängen einer komplett neuen Abfrage gefährdet. Aber der Angreifer könnte auch mit der `SET` Klausel herumspielen. In diesem Fall muss eine Schemainformation vorhanden sein, um die Abfrage erfolgreich manipulieren zu können. Diese kann durch Untersuchen der Variablennamen im Formular, oder simpel mittels brute force gesammelt werden. Es gibt nicht so viele Namenskonventionen für Felder, welche Passwörter oder Benutzernamen speichern.

Beispiel 4-8. Vom Zurücksetzen eines Passwortes ... zum Erlangen von mehr Rechten (irgendein Datenbankserver)

```
$query = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";
```

Aber ein böswilliger Benutzer übermittelt den Wert `' or uid like'%admin%'; --` zu `$uid`, um das Administrator Passwort zu ändern, oder setzt einfach `$pwd` auf `"hehehe', admin='yes', trusted=100 "` (mit dem hinteren Leerzeichen), um mehr Rechte zu erhalten. Dann wird die Abfrage verdreht:

```
// $uid == ' or uid like'%admin%'; --
$query = "UPDATE usertable SET pwd='...' WHERE uid=" or uid like '%admin%'; -
-";

// $pwd == "hehehe', admin='yes', trusted=100 "
$query = "UPDATE usertable SET pwd='hehehe', admin='yes', trusted=100 WHERE ...;"
```

Ein furchterregendes Beispiel, wie der Zugriff auf Kommandos auf Betriebssystemebene bei manchen Datenbankservern erfolgen kann.

Beispiel 4-9. Angriff auf das Betriebssystem des Datenbank Hosts (MSSQL Server)

```
$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";
$result = mssql_query($query);
```

Wenn ein Angreifer den Wert `a%' exec master..xp_cmdshell 'net user test testpass /ADD' --` zu `$prod` überträgt, wird `$query` zu:

```
$query  = "SELECT * FROM products
                    WHERE id LIKE '%a%'
                    exec master..xp_cmdshell 'net user test testpass /ADD'-
-";
$result = mssql_query($query);
```

Der MSSQL Server führt die SQL Statements in dem Batch aus, inklusive einem Kommando zum Anlegen eines neuen Benutzers in der Datenbank Accounts. Würde diese Applikation als `sa` und der MSSQLSERVER Service mit genügend Rechten laufen, hätte der Angreifer nun ein Konto, mit welchem er Zugriff auf diese Maschine hätte.

> **Anmerkung:** Manche der obigen Beispiele sind an einen spezifischen Datenbankserver gebunden. Das heißt jedoch nicht, dass nicht ein ähnlicher Angriff auf andere Produkte möglich wäre. Ihr Datenbankserver könnte auf andere Weise genauso verwundbar sein.

Techniken zur Vermeidung

Sie könnten sich nun darauf berufen, dass der Angreifer in den meisten Beispielen ein Stück Information über das Datenbankschema haben muss. Sie haben recht, aber Sie wissen nie, wann und wie es genommen werden kann, und wenn es passiert, kann Ihre Datenbank entblößt werden. Wenn Sie ein Open Source, oder öffentlich verfügbares Paket zur Handhabung von Datenbanken verwenden, welches vielleicht zu einem Content Management System oder Forum gehört, können Eindringlinge leicht eine Kopie eines Stücks Ihres Codes erstellen. Es kann auch ein Sicherheitsrisiko sein, wenn es sich um ein schlecht designtes Paket handelt.

Diese Angriffe basieren hauptsächlich auf dem Ausnutzen des Codes, welcher ohne Bedenken auf die Sicherheit geschrieben wurde. Vertrauen Sie nie auf irgendeine Art von Input, speziell wenn er von der Clientseite kommt, selbst wenn er von einer Auswahlbox, einem versteckten Eingabefeld, oder einem Cookie kommt. Das erste Beispiel zeigt, dass solch eine untadelige Abfrage ein Disaster anrichten kann.

- Stellen Sie nie als Superuser oder Owner einer Datenbank eine Verbindung zur Datenbank her. Verwenden Sie immer speziell angelegte Benutzer mit sehr limitierten Rechten.
- Prüfen Sie, ob der gegebene Input dem erwarteten Datentyp entspricht. PHP bietet eine große Auswahl an Funktionen zum Validieren des Input, von den einfachsten unter Variablenfunktionen und Character Type Functions (z.B. is_numeric() bzw. ctype_digit()), bis hin zu den Perl kompatiblen Regulären Ausdrücken.
- Wenn die Applikation numerischen Input erwartet, erwägen Sie die Prüfung der Daten mit is_numeric(), oder die Änderung des Typs mit settype(), oder verwenden Sie die numerische Repräsentation mittels sprintf().

Beispiel 4-10. Ein sicherer Weg, eine Abfrage zu erstellen

```
settype($offset, 'integer');
$query = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

// Beachten Sie %d im Formatstring, %s zu verwenden wäre sinnlos
$query = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20 OFF-
SET %d;",
                 $offset);
```

- Escapen Sie jeden nicht numerischen Input, welcher zur Datenbank weitergereicht werden soll mit addslashes() oder addcslashes(). Siehe auch das erste Beispiel. Wie dieses Beispiel zeigt, sind

in den statischen Teil der Abfrage eingebrachten Escapes nicht genug, und können leicht gehacked werden.

- Geben Sie keinerlei datenbankspezifische Informationen aus, speziell über das Schema, egal wie (auf ehrliche oder unehrliche Weise). Siehe auch Fehlerbehandlung und Error Handling and Logging Functions.
- Sie können stored procedures und vorher definierte Cursor verwenden, um den Datenzugriff zu abstrahieren, sodass Benutzer nicht direkt auf Tabellen oder Views zugreifen, aber diese Lösung hat andere Auswirkungen.

Abgesehen davon profitieren Sie von einer Protokollierung der Abfragen entweder in Ihrem Skript, oder durch die Datenbank selbst, wenn es hilft. Klar, die Protokollierung kann nicht irgendeinen schädlichen Versuch verhindern, aber es kann helfen herauszufinden, welche Applikation umgangen wurde. Das Log ist durch die in ihm enthaltene Information nützlich, und je mehr Details es enthält, desto besser ist es im Allgemeinen.

Fehlerbehandlung

PHP Security hat zwei Seiten der Fehlerbehandlung. Eine ist für die Erhöhung der Sicherheit vorteilhaft, die andere ist schädlich.

Eine Standard-Angriffstaktik beinhaltet die Erstellung eines Profils des anzugreifenden Systems, indem die aufgrund der Einspeisung von unzulässigen Daten zurückgegebenen Fehlermeldungen anhand deren Art und des Kontextes ausgewertet werden. Wenn z.B. ein Angreifer Informationen über eine auf einem eingesendeten Formular basierte Seite zusammengetragen hat, kann er versuchen, Variablen zu überschreiben bzw. zu modifizieren:

Beispiel 4-11. Variablen mit einer eigenen HTML Seite angreifen

```
<form method="post" action="attacktarget?username=badfoo&password=badfoo">
<input type="hidden" name="username" value="badfoo">
<input type="hidden" name="password" value="badfoo">
</form>
```

Die normalerweise zurückgegebenen PHP Fehler können für den Entwickler hilfreich sein, wenn dieser ein Skript debuggen möchte, Hinweise auf eine nicht korrekt arbeitende Funktion oder Datei, oder die PHP Datei und die Zeilennummer des aufgetretenen Fehlers anzeigen lassen möchte. Das sind alles Informationen, die ausgenutzt werden können. Es ist für einen PHP Entwickler nicht unüblich, show_source(), highlight_string(), oder highlight_file() zur Fehlersuche zu verwenden, jedoch kann dies in einem lebenden System auch versteckte Variablen, ungeprüfte Syntax und andere gefährliche Informationen aufdecken. Speziell gefährlich ist es, Code von bekannten Quellen mit integrierten Debugging Handlern auszuführen, oder weit verbreitete Debuggingtechniken zu verwenden. Wenn ein Angreifer die von Ihnen benutzte generelle Technik herausfindet, kann er versuchen, mit Brute-Force Ihre Seite zu knacken, indem er verschiedene allgemein gebräuchliche Debug Strings sendet:

Beispiel 4-12. Ausnutzen von gebräuchlichen Debugging Variablen

```
<form method="post" action="attacktarget?errors=Y&amp;showerrors=1"&debug=1">
<input type="hidden" name="errors" value="Y">
<input type="hidden" name="showerrors" value="1">
<input type="hidden" name="debug" value="1">
</form>
```

Ungeachtet der Fehlerbehandlungsmethode führt die Möglichkeit ein System nach Fehlermeldungen sondieren dazu, dass einem Angreifer mehr Informationen geboten werden.

Zum Beispiel weist schon alleine der Stil einer Fehlermeldung darauf hin, dass auf einem System PHP läuft. Wenn der Angreifer auf eine .html Seite kommt und untersuchen möchte welches System im Hintergrund läuft (um nach bekannten Systemschwächen zu suchen), könnte dieser mittels der Einspeisung von falschen Daten herausfinden, dass ein System mit PHP aufgebaut ist.

Ein Fehler einer Funktion gibt Aufschluss darüber, ob ein System eine bestimmte Datenbankapplikation benutzt, oder gibt Hinweise darauf, wie eine Webseite programmiert bzw. entworfen wurde. Dies erlaubt eine tiefere Überprüfung von offenen Datenbank-Ports, oder die Suche nach spezifischen Bugs bzw. Schwächen einer Webseite. Mit der Einspeisung von falschen Daten kann ein Angreifer z.B. die Reihenfolge der Authentifizierung in einem Skript bestimmen (anhand der Zeilennummern in den Fehlermeldungen), wie auch durch "Herumstochern" Missbrauchsmöglichkeiten an verschiedenen Stellen im Script herausfinden.

Eine Fehlermeldung des Dateisystems oder eines generellen PHP-Errors welche Rechte der Server hat, wie auch die Struktur und Organisation der Dateien auf dem Webserver. Vom Entwickler geschriebene Fehlermeldungen können das Problem verschlimmern, bis hin zum Preisgeben von zuvor "versteckten" Informationen.

Es gibt drei bedeutende Lösungen zu diesem Thema. Die erste ist, alle Funktionen zu überprüfen und zu versuchen, die Menge an Fehlermeldungen zu ersetzen. Die zweite ist, die Ausgabe von Fehlermeldungen am laufenden Code generell zu deaktivieren. Die dritte ist, sich unter Verwendung der PHP Funktionen zur Fehlerbehandlung seinen eigenen Error-handler zu schreiben. Abhängig von Ihrer Sicherheitspolitik könnte jede der drei Lösungen für Sie geeignet sein.

Ein Weg, diesen Punkt vorzeitig zu behandeln ist, das PHP eigene error_reporting() zu benutzen, um Ihren Code sicherer zu gestalten und möglicherweise gefährliche Nutzungen von Variablen zu entdecken. Wenn Sie Ihren Code noch vor dem Einsatz mit E_ALL testen, können Bereiche entdecken, in denen Ihre Variablen eventuell für Verseuchung oder andere Modifikationen offen sind. Sind Sie bereit zum Einsatz, können Sie Ihren Code mit E_NONE vor Sondierungen schützen.

Beispiel 4-13. Gefährliche Variablen mit E_ALL finden

```
<?php
if ($username) {  // Vor Verwendung nicht initialisiert oder geprüft
    $good_login = 1;
}
if ($good_login == 1) { // Wenn der obige Test fehlschlägt, ist vor der
                        // Verwendung nicht initialisiert oder geprüft
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Verwendung von Register Globals

Ein Feature von PHP zur Erhöhung der Sicherheit ist die Konfiguration von PHP mit register_globals = off. Mit Deaktivierung der Möglichkeit, irgendeine vom Benutzer übertragenen Variable in den PHP Code zu injizieren, können Sie die Anzahl "vergifteter" Variablen reduzieren, welche ein potentieller Angreifer zufügen könnte. Dieser benötigt mehr Zeit, um sich Übermittlungen auszudenken, und Ihre internen Variablen sind effektiv von den übergebenen Benutzervariablen isoliert.

Während dies den benötigten Aufwand mit PHP zu arbeiten leicht erhöht ist dargelegt, dass die Vorteile gegenüber dem Aufwand klar überwiegen.

Beispiel 4-14. Ohne register_globals=off arbeiten

```
<?php
if ($username) {  // kann vom User mit get/post/cookies übermittelt werden
    $good_login = 1;
}

if ($good_login == 1) { // kann vom User mit get/post/cookies übermittelt werden
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Beispiel 4-15. Mit register_globals = off arbeiten

```
<?php
if($HTTP_COOKIE_VARS['username']){
    // kann nur von einem Cookie kommen
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Dies weise genutzt ist es auch möglich, präventive Messungen durchzuführen, um bei versuchten Vorstößen zu warnen. Wenn Sie im Voraus wissen, woher eine Variable kommen soll, können Sie prüfen, ob die übermittelten Daten nicht einen unpassenden Weg genommen haben. Obwohl dies nicht garantiert, dass Daten nicht nur ausgedacht sind, erfordert es von einem Angreifer, auch den richtigen Weg zu finden.

Beispiel 4-16. Entdecken einfacher Manipulationen von Variablen

```
<?php
if ($HTTP_COOKIE_VARS['username'] &&
    !$HTTP_POST_VARS['username'] &&
    !$HTTP_GET_VARS['username'] ) {
    // Durchführen anderer Checks, ob der Benutzername gültig ist...
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
} else {
   mail("admin@example.com", "Possible breakin attempt", $HTTP_SERVER_VARS['REMOTE_ADDR']
   echo "Security violation, admin has been alerted.";
   exit;
}
?>
```

Natürlich bedeutet ein einfaches Deaktivieren von register_globals nicht, dass Ihr Code nun automatisch sicher ist. Jeder Teil mit Daten sollte auch auf andere Arten geprüft werden.

Vom Nutzer übermittelte Daten

In vielen PHP Programmen liegt die größte Schwäche nicht an der Sprache selbst, sondern bloß an Code, der nicht mit dem nötigen Augenmerk auf die Sicherheit geschrieben wurde. Deshalb sollten Sie sich immer Zeit nehmen, die Implikationen eines gegebenen Codestücks zu bedenken, um einen möglichen Schaden durch eine eventuell unerwartete übergebene Variable festzustellen.

Beispiel 4-17. Gefährliche Verwendung von Variablen

```
<?php
// lösche eine Datei aus dem Benutzer-Verzeichnis...
// oder vielleicht dem eines anderen Benutzers?
unlink ($evil_var);

// Schreibe die Log-Information von deren Zugriff...
// oder vielleicht einen /etc/password Eintrag?
fputs ($fp, $evil_var);

// Führe etwas triviales aus... oder rm -rf *?
system ($evil_var);
exec ($evil_var);

?>
```

Sie sollten Ihren Code immer sorgfältig kontrollieren, um eine sorgfältige Prüfung irgendwelcher von einem Web-Browser übertragenen Variablen sicherzustellen, und sich selbst folgende Fragen stellen:

- Wird sich dieses Skript nur auf die vorgesehenen Dateien auswirken?
- Kann auf ungewöhnliche oder unerwünschte Daten reagiert werden?
- Kann dieses Skript auf nicht vorgesehene Art genutzt werden?
- Kann dies in Verbindung mit anderen Skripten in einer negativen Art benutzt werden?
- Werden alle Transaktionen ausreichend geloggt?

Wenn Sie sich diese Fragen anstatt danach schon während dem Schreiben des Skriptes stellen, ersparen Sie sich das unangenehme Umschreiben, wenn eine Erhöhung der Sicherheit erforderlich wird. Mit dieser Art zu denken werden Sie die Sicherheit des Systems zwar nicht garantieren, aber sie können helfen, sie zu erhöhen.

Sie könnten auch Überlegungen anstellen, ob Sie nicht register_globals, magic_quotes, oder andere bequeme Einstellungen abschalten, welche Sie in Punkto Gültigkeit, Herkunft, oder Inhalt einer gegebenen Variable durcheinanderbringen könnten. Mit PHP im error_reporting(E_ALL) Modus zu arbeiten kann auch helfen, Sie vor Variablen zu warnen, welche benutzt werden, bevor sie geprüft oder initialisiert wurden (so können Sie verhindern, dass mit ungewöhnlichen Daten gearbeitet wird).

Verstecken von PHP

Generell ist Sicherheit durch Unklarheit eine der schwächsten Formen von Sicherheit. Aber in manchen Fällen ist ein klein Wenig mehr an extra Sicherheit wünschenswert.

Ein paar einfache Techniken helfen PHP zu Verstecken, um einen nach Schwächen in Ihrem System suchenden Angreifer möglicherweise langsamer Wenn Sie in Ihrer php.ini expose_php = off zu machen. setzen, reduzieren Sie damit die ihm zur Verfügung stehenden Informationen.

Eine andere Taktik ist, den Webserver wie z.B. Apache entweder mittels einer .htaccess Direktive oder in der Apache Konfigurationsdatei selbst so konfigurieren, dass dieser verschiedene Dateitypen durch PHP parst. So können Sie irreführende Dateierweiterungen verwenden:

Beispiel 4-18. PHP als andere Sprache ausgeben

```
# Lasse PHP Code wie andere Arten von Code aussehen
AddType application/x-httpd-php .asp .py .pl
```

Oder komplett unklar machen:

Beispiel 4-19. Verwenden von unbekannten Typen für PHP Dateierweiterungen

```
# Lasse PHP Code wie unbekannte Typen aussehen
AddType application/x-httpd-php .bop .foo .133t
```

Oder verstecken Sie ihn als html Code, was einen leichten Performanceverlust bedeutet, da alle html Dateien durch die PHP Engine geparst werden:

Beispiel 4-20. Verwenden von html Typen für PHP Dateierweiterungen

```
# Lasse PHP code wie html aussehen
AddType application/x-httpd-php .htm .html
```

Um dies effektiv arbeiten zu lassen, müssen Sie Ihre PHP Dateien nach den obigen Dateierweiterungen umbenennen. Während dies eine Form der Sicherheit durch Verhüllung ist, ist es ein kleines präventives Maß mit ein paar Nachteilen.

Aktuell bleiben

PHP wird, wie jedes andere große System auch, ständig geprüft und weiterentwickelt. Jede neue Version wird oft sowohl große, als auch kleine Änderungen enthalten, welche die Gesamtsicherheit und Stabilität des Systems betreffenden Punkte (z.B. Sicherheit, Konfiguration) verbessern bzw. vorhandene Mängel beheben.

Wie auch bei anderen Scriptsprachen und Programmen auf Systemebene ist der beste Ansatz ein oftmaliges Update, und sich laufend über die letzten Versionen und deren Änderungen zu informieren.

Teil II. Sprachreferenz

Kapitel 5. Grundlagen der Syntax

Den HTML-Bereich der Datei verlassen

Es gibt vier Möglichkeiten, den HTML-Bereich einer Skript-Datei zu verlassen und in den "PHP-Modus" zu wechseln:

Beispiel 5-1. Möglichkeiten, den HTML-Bereich zu verlassen

```
1.  <? echo ("Das ist die einfachste Möglichkeit: eine SGML-´proccessing in-
struction´\n"); ?>

2.  <?php echo("Um XML-konforme Dokumente herzustellen, benutzt man diese Syn-
tax\n"); ?>

3.  <script language="php">
        echo ("Manche Editoren (z. B. Frontpage) mögen keine ´processing instructions´\n'
    </script>

4.  <% echo ("Optional können auch Tags im ASP-Stil verwendet werden"); %>
    <%= $variable; # Das ist ein Abkuerzung fuer "<%echo .." %>
```

Die erste Methode ist nur verfügbar, wenn ´short tags´ aktiviert sind. Dies kann durch die **short_tags()**-Funktion geschehen, durch Aktivieren der short_open_tag- Einstellung in der PHP-Konfigurationsdatei oder durch das Übergeben der ´--enable-short-tags´-Option an den **configure**-Befehl beim Kompilieren des PHP-Parsers.

Die vierte Methode ist nur verfügbar, wenn Tags im ASP-Stil (ASP-style tags) aktiviert sind. Das ist möglich, indem man die asp_tags-Konfigurationseinstellung aktiviert.

> **Anmerkung:** Die Unterstützung der Tags im ASP-Stil wurde in Version 3.0.4. hinzugefügt.

Der schließende Tag für den PHP-Block schließt eine sofort folgende Zeilenschaltung mit ein, falls diese vorhanden ist. Außerdem beinhaltet der schließende Tag ein implizites Semikolon; Sie müssen also die letzte Zeile eines PHP-Blocks nicht mit einem Semikolon beenden.

Abgrenzung von Anweisungen

Einzelne Anweisungen werden genauso voneinander abgetrennt wie in C oder Perl - jedes Statement wird mit einem Semikolon beendet.

Auch der schließende Tag (?>) bedeutet das Ende eines Statements, deshalb sind die folgenden Beispiele gleichwertig:

```
<?php
    echo "Dies ist ein Test";
?>

<?php echo "Dies ist ein Test" ?>
```

Kommentare

PHP unterstützt ’C’, ’C++’ und Unix-Shell-artige Kommentare. Beispiele:

```
<?php
    echo "Dies ist ein Test"; // Dies ist ein einzeiliger Kommentar im C++-
Stil.
    /* Dies ist ein mehrzeiliger Kommentar
       und noch eine Kommentar-Zeile */
    echo "Dies ist noch ein Test";
    echo "... und ein abschliessender Test"; # Dies ist ein Shell-artiger Kommentar.
?>
```

Die einzeiligen Kommentar-Arten kommentieren sämtlichen Text bis zum Zeilenende oder bis zum Ende des aktuellen PHP-Blocks aus, je nachdem, was zuerst eintritt.

```
<h1>Dies ist ein  <?# echo "einfaches";?> Beispiel.</h1>
<p>Obige Überschrift wird lauten: ’Dies ist ein Beispiel.’.
```

Man sollte darauf achten, nicht mehrere ´C´-artigen Kommentare ineinander zu setzen, was vorkommen kann, wenn man große Blöcke auskommentiert.

```
<?php
 /*
    echo "Dies ist ein Test"; /* Dieser Kommentar wird ein Problem verursachen. */
 */
?>
```

Kapitel 6. Typen

Einführung

PHP unterstützt acht primitive Typen.

Vier skalare Typen:

- Boolean
- Integer
- Fließkomma-Zahl (float)
- String / Zeichenkette

Zwei zusammengesetzte Typen:

- Array
- Object

Und zuletzt zwei spezielle Typen:

- Resource
- NULL

> **Anmerkung:** In diesem Manual werden Sie oft `mixed` Parameter finden. Diese Pseudo-Typen weisen darauf hin, dass es mehrere Möglichkeiten für diesen Parameter gibt.

Der Typ einer Variabalen wird normalerweise nicht vom Programmierer bestimmt; vielmehr wird dies zur Laufzeit von PHP entschieden, abhängig vom Zusammenhang in dem die Variable benutzt wird.

> **Tipp:** Um den Typ und den Wert eines bestimmten Ausdrucks (Expression) zu überprüfen, können Sie var_dump() benutzen.
>
> Wenn Sie zur Fehlersuche einfach nur eine lesbare Darstellung eines Typs benötigen, benutzen Sie gettype(). Um auf einen bestimmten Typ zu prüfen, sollten Sie *nicht* gettype() benutzen. Stattdessen sollten Sie die `is_type` Funktionen verwenden.

Wenn sie die Umwandlung in einen bestimmten Variablen-Typ erzwingen wollen, erreichen Sie dies entweder durch cast oder durch Gebrauch der Funktion settype().

Beachten Sie, dass sich eine Variable in bestimmten Situationen unterschiedlich verhalten kann, abhängig vom Typ dem die Variable zu dem Zeitpunkt entspricht. Weitere Informationen entnehmen Sie dem Abschnitt zur Typ-Veränderung.

Boolscher Typ

Das ist der einfachste Typ. Ein boolean drückt einen Wahrheitswert aus. Dieser kann entweder `TRUE` oder `FALSE` sein.

> **Anmerkung:** Der boolsche Typ wurde in PHP 4 eingeführt.

Syntax

Um einen boolschen Typ wörtlich anzugeben, benutzen Sie entweder das Schlüsselwort `TRUE` oder `FALSE`. Beide unterscheiden sich nicht bezüglich der Groß- und Kleinschreibung.

```
$foo = True; // $foo wird der Wert TRUE zugewiesen
```

Üblicherweise verwenden Sie einen der Operatoren, der Ihnen einen boolean Wert liefert, den Sie dann an eine der Kontroll-Strukturen weiterreichen.

```
// == ist ein Vergleichs-Operator der einen boolschen Wert liefert
if ($action == "show_version") {
    echo "Die Version ist 1.23";
}

// Das ist nicht notwendig:
if ($show_separators == TRUE) {
    echo "<hr>\n";
}

// weil Sie einfach folgendes schreiben können:
if ($show_separators) {
    echo "<hr>\n";
}
```

Umwandlung nach boolean

Um einen Wert ausdrücklich nach boolean zu konvertieren benutzen Sie entweder die Umwandlung mittels `(bool)` oder `(boolean)`. In den allermeisten Fällen ist es jedoch nicht notwendig die Umwandlung selbst vorzunehmen. Ein Wert wird automatisch konvertiert, falls ein Operator, eine Funktion oder eine Kontrollstruktur ein boolean Argument erfordert.

Siehe auch Typ-Veränderung.

Bei der Umwandlung nach boolean werden folgende Werte als `FALSE` angesehen:

- das boolean `FALSE`
- die Integer 0 (Null)
- die Fließkomma-Zahl 0.0 (Null)
- die leere Zeichenkette und die Zeichenkette "0"

- ein Array ohne Elemente
- ein Object ohne Elemente
- der spezielle Type NULL (einschließlich nicht definierter Variablen)

Jeder andere Wert wird als `TRUE` angesehen (einschließlich jeder Resource).

Warnung

`-1` wird als `TRUE` angesehen, wie jede andere Zahl ungleich Null (egal ob negativ oder positiv)!

Integer Typen

Ein Integer ist eine Nummer aus der Menge Z = {..., -2, -1, 0, 1, 2, ...}.

Siehe auch: Beleibig präzise Ganzzahlen und Fließkomma-Zahlen

Syntax

Ganzzahlen können in dezimaler (10-basierter), hexadezimaler (16-basierter) oder oktaler (8-basierter) Schreibweise angegeben werden, wahlweise mit einem vorangestellten Zeichen (- oder +).

Wenn Sie die oktale Schreibweise verwenden, müssen Sie die Zahl mit einer vorangestellten Null `0` schreiben; in hexadezimaler Schreibweise müssen Sie vor die Zahl ein `0x` schreiben.

Beispiel 6-1. Integer Literale

```
$a = 1234; // Dezimalzahl
$a = -123; // eine negative Zahl
$a = 0123; // Oktalzahl (entspricht 83 dezimal)
$a = 0x1A; // Hexadezimalzahl (entspricht 26 dezimal)
```

Die Größe eines Integer-Wertes ist plattformabhängig, ein Maximalwert von ungefähr zwei Milliarden ist jedoch üblich. (Vorzeichen-behafteter 32-Bit-Wert). PHP unterstützt keine vorzeichenlose Integer-Werte.

Integer Überlauf

Wenn Sie eine Zahl jenseits der Grenzen des Typs Integer angeben, wird diese stattdessen als Typ float interpretiert. Wenn Sie eine Operation ausführen, deren Ergebnis eine Zahl jenseits der Grenzen des Typs Integer ist, wird ebenso eine Zahl vom Typ float zurückgegeben.

```
$large_number =  2147483647;
```

```
var_dump($large_number);
// Ausgabe: int(2147483647)

$large_number =  2147483648;
var_dump($large_number);
// Ausgabe: float(2147483648)

// das gilt auch für Integers in hexadezimaler Schreibweise:
var_dump( 0x80000000 );
// Ausgabe: float(2147483648)

$million = 1000000;
$large_number =  50000 * $million;
var_dump($large_number);
// Ausgabe: float(50000000000)
```

Warnung

Bedauerlicherweise gab es einen Bug in PHP, der die korrekte Funktionsweise verhinderte, wenn negative Zahlen verwendet wurden. Ein Beispiel: Bei der Ausführung von `-50000 * $million` war das Ergebnis `-429496728`. Sind beide Operanden positiv gibt gibt es keine Probleme.

Dieses Problem ist in PHP 4.1.0 behoben.

In PHP gibt es keinen Operator für Integer Divisonen. `1/2` ergibt float `0.5`.

```
var_dump( 25/7 );
// Ausgabe: float(3.5714285714286)
```

Umwandlung nach Integer

Um einen Wert ausdrücklich nach integer zu konvertieren benutzen Sie entweder die Umwandlung mittels `(int)` oder `(integer)`. In den allermeisten Fällen ist es jedoch nicht notwendig die Umwandlung selbst vorzunehmen. Ein Wert wird automatisch konvertiert, falls ein Operator, eine Funktion oder eine Kontrollstruktur ein integer Argument erfordert.

Siehe auch Typ-Veränderung.

Von Booleans

`FALSE` ergibt `0` (Null), und `TRUE` ergibt `1` (eins).

Von Fließkomma-Zahlen

Bei der Umwandlung von float nach integer wird die Zahl *in Richtung Null* gerundet.

Wenn der float jenseits der Grenzen von integer liegt (üblicherweise `+/- 2.15e+9 = 2^31`), ist das Ergebnis nicht definiert, weil float nicht genug Präzision besitzt um ein genaues integer Ergebnis zu liefern. Keine Warnung, nicht einmal eine Notiz wird in diesem Fall ausgegeben.

Warnung

Wandeln Sie nie ein Teil eines Ausdrucks nach integer um, da dies manchmal zu unerwarteten Ergebnissen führen kann.

```
echo (int) ( (0.1+0.7) * 10 ); // Ausgabe: 7!
```

Für mehr Informationen schauen Sie unter Warnung über float-Präzision nach.

Von Zeichenketten/ Strings

Siehe String Umwandlung

Von anderen Typen

Achtung

Das Verhalten bei der Umwandlung nach integer ist für andere Typen nicht definiert. Zum gegenwärtigen Zeitpunkt ist das Verhalten so, als ob der Wert zuerst nach boolean konvertiert wird. Auf jeden Fall sollten Sie sich auf dieses Verhalten *nicht* verlassen. Es kann sich ohne Ankündigung ändern.

Fließkomma-Zahlenwerte

Fließkomma-Zahlenwerte ("floats", "doubles" oder "reelle Zahlen") können durch eine der folgenden Anweisungen zugewiesen werden:

```
$a = 1.234;
$a = 1.2e3;
$a = 7E-10;
```

Die Größe einer Fließkomma-Zahl ist plattformabhängig, dennoch stellt ein Maximum von ~1.8e308 mit einer Genauigkeit von 14 Nachkomma-Stellen einen üblichen Wert dar (das ist 64-Bit im IEEE-Format).

Fließkomma Päzision

Es ist ziemlich normal, dass einfache Dezimalzahlen wie `0.1` oder `0.7` nicht in ihre internen binären Entsprechungen konvertiert werden können, ohne einen kleinen Teil ihrer Genauigkeit zu verlieren. Das kann zu verwirrenden Ergebnissen führen. So wird `floor((0.1 + 0.7) * 10)` normalerweise `7` statt des erwarteten Wertes `8` zurück geben (als Ergebnis der internen Entsprechung von `7.9999999999...`

Das gründet sich auf die Tatsache, dass es unmöglich ist, manche Dezimal-Zahlen durch eine endliche Anzahl an Nachkomma-Stellen darzustellen. Dem Wert `1/3` entspricht z.B. der interne Wert von `0.3333333...`

Deshalb sollten sie nie den Ergebnissen von Fließkomma-Operationen bis auf die letzte Nachkomma-Stelle trauen und nie solche auf Gleichheit prüfen. Benötigen Sie wirklich eine größere Genauigkeit, sollten sie die mathematischen Funktionen beliebiger Genauigkeit oder die Gmp Funktionen benutzen.

Strings / Zeichenketten

Ein string ist eine Folge von Zeichen. In PHP entspricht ein Zeichen einem byte, das heißt, dass exakt 256 verschiedene Zeichen möglich sind. Das impliziert auch, dass PHP keine native Unterstützung für Unicode bietet.

> **Anmerkung:** Für einen String stellt die Länge kein Problem dar. Von PHP-Seite aus gibt es keine praktische Grenze für die Größe eines Strings. Daher gibt es keinen Grund sich Sorgen über lange Strings zu machen.

Syntax

Ein String kann auf drei verschiedene Weisen geschrieben werden.

- Einfache Anführungszeichen (single quoted)
- Doppelte Anführungszeichen (double quoted)
- Heredoc Syntax

Einfache Anführungszeichen (Single quoted)

Der leichteste Weg einen einfachen String zu schreiben, ist der Einschluss in einfache Anführungszeichen (das Zeichen `'`).

Um ein einfaches Anführungszeichen wörtlich auszugeben, muss dieses mit einem Backslash (`\`) escaped werden, wie in vielen anderen Programmiersprachen auch. Wenn Sie innerhalb eines Strings einen Backslash vor einem einfachen Anführungszeichen oder am Ende eines Strings ausgeben wollen, müssen Sie diesen verdoppeln. Beachten Sie: wenn Sie versuchen irgendwelche anderen

Zeichen zu escapen, wird der Backslash ebenfalls ausgegeben! Daher besteht für gewöhnlich keine Notwendigkeit den Backslash selbst zu escapen.

Anmerkung: In PHP 3 wird eine Warnung auf der Stufe `E_NOTICE` ausgegeben, wenn das passiert.

Anmerkung: Anders als bei den zwei anderen Schreibweisen werden Variablen innerhalb von single-quoted Strings *nicht* ausgewertet.

```
echo 'Das ist ein einfacher String';
echo 'Sie können ebenso einen Zeilenumbruch einfügen,
auf diese Art.';

echo 'Arnold sagte einmal: "I\'ll be back"';
// Ausgabe: ... "I'll be back"

echo 'Sind Sie sicher, dass Sie C:\\*.* löschen wollen?';
// Ausgabe: ... dass Sie C:\*.* löschen wollen?

echo 'Sind Sie sicher, dass Sie C:\*.* löschen wollen?';
// Ausgabe: ... dass Sie C:\*.* löschen wollen?

echo 'Ich versuche einen Zeilenumbruch an diesem Punkt: \n ein Zeilenumbruch';
// Ausgabe: ... diesem Punkt: \n ein Zeilenumbruch'

$var = "nur zum Vergnügen";
echo 'Das mache ich $var';
// Ausgabe: Das mache ich $var
```

Doppelte Anführungszeichen (Double quoted)

Wenn ein String in doppelten Anführungszeichen (") eingeschlossen ist, versteht PHP mehr Escape-Folgen für spezielle Zeichen:

Tabelle 6-1. Nicht ausgewertete / übergangene Zeichen:

Zeichenfolge	Bedeutung
`\n`	Zeilenvorschub (LF oder 0x0A als ASCII-Code)
`\r`	Wagenrücklauf (CR oder 0x0D als ASCII-Code)
`\t`	horizontaler Tabulator (HT oder 0x09 als ASCII-Code
`\\`	Backslash / Rückstrich
`\$`	Dollar-Symbol

Zeichenfolge	Bedeutung
`\"`	doppelte Anführungszeichen
`\[0-7]{1,3}`	die Zeichenfolge, die dem regulären Ausdruck entspricht ist ein Zeichen in Oktal-Schreibweise
`\x[0-9A-Fa-f]{1,2}`	die Zeichenfolge, die dem regulären Ausdruck entspricht ist ein Zeichen in Hexadezimal-Schreibweise

Noch einmal: wenn Sie versuchen, irgend ein anderes Zeichen zu escapen wird der Backslash ebenfalls ausgegeben!

Der wichtigste Vorteil von double-quoted Strings ist die Tatsache, dass Variablennamen ausgewertet werden. Für Details siehe String Analyse (parsing)

Heredoc

Eine andere Möglichkeit Strings einzufassen, besteht im Gebrauch der heredoc-Syntax ("<<<"). Hierfür ist nach `<<<` ein Bezeichner zu setzen. Nun folgt der eigentliche String und dann derselbe Bezeichner um den String abzuschliessen.

Der schliessende Bezeichner *muss* in der ersten Spalte der Zeile stehen. Die verwendeten Bezeichner müssen den gleichen Regeln entsprechen wie alle anderen PHP-Labels auch: Sie dürfen lediglich alphanumerische Zeichen und den Unterstrich enthalten und müssen mit einem Unterstrich oder einem Buchstaben beginnen.

Warnung

Es ist sehr wichtig zu beachten, dass die Zeile mit dem schliessenden Bezeichner keine anderen Zeichen enthält, ausgenommen *möglicherweise* ein Semikolon (`;`). Das bedeuted im Besonderen, dass der Bezeichner *nicht eingerückt werden darf* und es dürfen keine Leerzeichen oder Tabulatoren vor oder nach dem Semikolon stehen.

Wahrscheinlich der unangenehmste Umstand ist der, dass ebenso kein Wagenrücklauf (`\r`) am Ende dieser Zeile stehen darf, nur ein Zeilenvorschub (`\n`). Da Microsoft Windows die Folge `\r\n` als Zeilenabschlusszeichen benutzt, kann Ihre heredoc-Syntax nicht funktionieren, wenn Sie Ihr Skript mit einem Editor unter Windows schreiben. Die meisten Programmier-Editoren bieten jedoch die Möglichkeit, Ihre Dateien mit einem UNIX Zeilenende zu speichern.

Heredoc-Text funktioniert wie ein String innnerhalb doppelter Anführungszeichen, nur ohne doppelte Anführungszeichen. Anführungszeichen innerhalb von heredoc-Texten müssen also keiner Sonderbehandlung (escapen) unterzogen werden, aber Sie können dennoch die oben aufgeführten Escape-Anweisungen verwenden. Variablen werden ausgewertet, aber besondere Aufmerksamkeit muss komplexen Variablen gewidmet werden, genau wie bei Strings.

Beispiel 6-2. Beispiel zur String-Festlegung per "heredoc"-Methode:

```
<?php
$str = <<<EOD
Beispiel eines Strings
über mehrere Script-Zeilen
```

```
durch Gebrauch der heredoc-Syntax.
EOD;

/* komplexeres Beispiel, mit Variablen */
class foo {
    var $foo;
    var $bar;

    function foo() {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'Mein Name';

echo <<<EOT
Mein Name ist "$name". Ich schreibe einige $foo->foo.
Nun schreibe ich gerade einige {$foo->bar[1]}.
Dies sollte ein großes 'A' schreiben: \x41
EOT;
?>
```

Anmerkung: Die heredoc Unterstützung wurde in PHP 4 eingeführt.

Variablen-Analyse (parsing)

Wird ein String in doppelten Anführungszeichen oder mit heredoc angegeben, werden enthaltene Variablen ausgewertet (geparst).

Es gibt zwei Syntax-Typen, eine einfache und eine komplexe. Die einfache Syntax ist die geläufigste und bequemste. Sie bietet die Möglichkeit eine Variable, einen Array-Wert oder eine Objekt-Eigenschaft auszuwerten (parsen).

Die komplexe Syntax wurde in PHP 4 eingeführt und ist an den geschweiften Klammern `{}`erkennbar, die den Ausdruck umschließen.

Einfache Syntax

Sobald ein Dollarzeichen (`$`) auftaucht, wird der Parser versuchen, gierig so viele Zeichen wie möglich zu nehmen, um einen gültigen Variablennamen zu bilden. Schließen Sie Ihren Varaiblennamen in geschweifte Klammern ein, wenn Sie ausdrücklich das Ende des Namens angeben wollen.

```
$beer = 'Heineken';
echo "$beer's Geschmack ist großartig";  // funktioniert, "'" ist kein gültiges
                                         // Zeichen für einen Variablennamen
echo "Er hat einige $beers getrunken";   // funktioniert nicht, 's' ist ein gültiges
```

```
                                        // Zeichen für einen Variablennamen
echo "Er hat einige ${beer}s getrunken"; // funktioniert
```

Auf ähnliche Weise können Sie erreichen, dass ein Array-Index oder eine Objekt-Eigenschaft ausgewertet wird. Bei Array-Indizes markiert die schließende eckige Klammer (`]`) das Ende des Index. Für Objekt-Eigenschaften gelten die gleichen Regeln wie bei einfachen Variablen, obwohl es bei Objekt-Eigenschaften keinen Trick gibt, wie dies bei Variablen der Fall ist.

```
$früchte = array( 'Erdbeere' => 'rot' , 'Banane' => 'gelb' );

// Beachte: außerhalb von String-Anführungszeichen funktioniert das anders.
echo "Eine Banane ist $früchte[Banane].";

echo "Dieses Quadrat ist $quadrat->breite Meter breit.";

// Funktioniert nicht. Für eine Lösung siehe die komplexe Syntax.
echo "Dieses Quadrat ist $quadrat->breite00 Zentimeter breit.";
```

Für irgendetwas Komplexeres sollten Sie die komplexe Syntax nutzen.

Komplexe (geschweifte) Syntax

Diese wird nicht komplex genannt, weil etwa die Syntax komplex ist, sondern weil Sie auf diesem Weg komplexe Ausdrücke einbeziehen können.

Tatsächlich können Sie jeden beliebigen Wert einbeziehen, der im Namensbereich in Strings gültig ist. Schreiben Sie den Ausdruck einfach auf die gleiche Art und Weise, wie außerhalb des Strings, und umschließen diesen mit '{' und '}'. Da Sie '{' nicht escapen können, wird diese Syntax nur erkannt, wenn auf '{' unmittelbar '$' folgt. (Benutzen Sie '{\$' oder '\{$' um ein wörtliches '{$' zu erhalten.) Einige Beispiele, um dies zu verdeutlichen:

```
$great = 'fantastisch';
echo "Das ist { $great}"; // funktioniert nicht, Ausgabe: Das ist { fantastisch}
echo "Das ist {$great}";  // funktioniert, Ausgabe: Das ist fantastisch
echo "Dieses Quadrat ist {$square->width}00 Zentimeter breit.";
echo "Das funktioniert: {$arr[4][3]}";

// Das ist aus dem gleichen Grund falsch
// wie $foo[bar] außerhalb eines Strings falsch ist.
echo "Das ist falsch: {$arr[foo][3]}";

echo "So sollten Sie es machen: {$arr['foo'][3]}";
echo "Sie können sogar schreiben {$obj->values[3]->name}";
echo "Das ist der Wert der Variable mit Namen $name: {${$name}}";
```

Zugriff auf Zeichen in Strings

Auf Zeichen innerhalb von Strings kann durch die Angabe des Null-basierten Offsets des gewünschten Zeichens, in geschweiften Klammern nach dem String, zugegriffen werden.

> **Anmerkung:** Für Abwärtskompatibilität können Sie immer noch die Array-Klammern verwenden. Diese Syntax wird jedoch seit PHP 4 missbilligt.

Beispiel 6-3. Einige String-Beispiele

```
<?php
// Zuweisung eines Strings.
$str = "Dies ist eine Zeichenkette";

// Erweiterung dieses Strings.
$str = $str . " mit etwas zusätzlichem Text";

// andere Möglichkeit zum Erweitern incl. Anweisung für Neue-Zeile.
$str .= " und einer neuen Zeile am Ende.\n";

/*
Nach diesen drei Operationen enthält $str:
"Dies ist eine Zeichenkette mit etwas zusätzlichem Text und einer neuen Zei-
le am Ende.\n."
*/

// Dieser String wird am Ende zu: '<p>Nummer: 9</p>'
$num = 9;
$str = "<p>Nummer: $num</p>";

// Dieser wird zu '<p>Nummer: $num</p>'
$num = 9;
$str = '<p>Nummer: $num</p>';

// Das erste Zeichen eines Strings.
$str = 'Das ist ein Test.'
$first = $str{0};                // $first enthält "D"

// Das letzte Zeichen eines Strings.
$str = 'Das ist immer noch ein Test.'
$last = $str{strlen($str)-1};    // $last enthält "."
?>
```

Nützliche Funktionen

Zeichenketten / Strings können mittels des '.'(dot)-Operators miteinander verbunden werden. Beachten Sie, dass in php dafür nicht der '+' (Additions)-Operator verwendet wird. Für mehr Informationen schauen Sie bitte unter den Zeichenketten-Operatoren nach.

Es gibt eine Menge nützlicher Funktionen zur String-Manipulation.

Für allgemeine Funktionen schauen Sie sich den Abschnitt über String Funktionen an, und für fortgeschrittenes Suchen & Ersetzen die Funktionen zu Regulären Ausdrücken (in zwei Ausführungen: Perl Kompatibel and POSIX erweitert).

Weiterhin gibt es auch noch Funktionen für URL-Strings, und Funktionen zum Verschlüsseln/Entschlüsseln von Strings (mcrypt und mhash).

Schließlich, falls Sie immer noch nicht das gefunden haben wonach Sie suchen, schauen Sie unter den Zeichen-Typen Funktionen nach.

Umwandlung von Zeichenketten / Strings

Sobald ein String als numerischer Wert ausgewertet wird, wird der resultierende Wert und Typ wie folgt festgelegt.

Der String wird als float ausgewertet, wenn er eines der Zeichen '.', 'e' oder 'E' enthält. Ansonsten wird er als Integer-Wert interpretiert.

Der Wert wird durch den Anfangsteil des Strings bestimmt. Sofern der String mit gültigen numerischen Daten beginnt, werden diese als Wert benutzt. Andernfalls wird der Wert 0 (Null) sein. Gültige numerische Daten sind ein optionales Vorzeichen, gefolgt von einer oder mehreren Zahlen (optional mit einem Dezimalpunkt). Wahlweise kann auch ein Exponent angegeben werden. Der Exponent besteht aus einem 'e' oder 'E', gefolgt von einer oder meheren Zahlen.

Sobald der erste Ausdruck ein String ist, hängt der Typ der Variablen vom zweiten Ausdruck ab.

```
$foo = 1 + "10.5";               // $foo ist float   (11.5)
$foo = 1 + "-1.3e3";             // $foo ist float   (-1299)
$foo = 1 + "bob-1.3e3";          // $foo ist integer (1)
$foo = 1 + "bob3";               // $foo ist integer (1)
$foo = 1 + "10 Kleine Schweine"; // $foo ist integer (11)
$foo = 1 + "10 Ferkel";          // $foo ist integer (11)
$foo = "10 Schweine " + 1;       // $foo ist integer (11)
$foo = "10.0 Schweine " + "1";   // $foo ist float   (11)
$foo = "10 Schweine " + 1.0;     // $foo ist float   (11)
```

Weitere Informationen über solche Umwandlungen finden Sie in den UNIX Manualseiten unter strtod(3).

Wenn Sie irgendein Beispiel dieses Abschnitts testen wollen, können Sie dieses per "Copy und Paste" übernehmen. Fügen Sie die folgende Zeile Ihrem Skript hinzu und Sie sehen selbst was passiert:

```
echo "\$foo==$foo; Typ ist " . gettype ($foo) . "<br>\n";
```

Arrays

Ein Array in PHP ist eigentlich eine geordnete Abbildung. Eine Abbildung ist ein Typ der *Werte* auf *Schlüssel* abbildet. Dieser Typ ist auf mehrere Arten optimiert, so dass Sie ihn auf vershiedene Weise benutzen können: als reales Array, als Liste (Vektor), als Hash-Tabelle (das ist eine praktische Anwendung einer Abbildung), als Verzeichnis, als Sammlung, als Stapel (Stack), als Warteschlange (Queue) und möglicherweise mehr. Da Sie ein weiteres PHP-Array als Wert benutzen können, ist es ziemlich einfach möglich Baumstrukturen zu simulieren.

Die Erklärung dieser Strukturen würde den Rahmen dieses Handbuchs sprengen, aber Sie finden zumindest ein Beispiel für jede dieser Strukturen. Für weitergehende Informationen verweisen wir auf externe Literatur zu diesem breit gefächerten Thema.

Syntax

Angabe mit array()

Ein array kann mit Hilfe des Sparchkonstrukts array() erzeugt werden. Es benötigt eine bestimmte Anzahl von Komma-getrennten `Schlüssel => Wert` Paaren.

Ein `Schlüssel` ist entweder eine nicht-negative Zahl vom Typ integer oder ein string. Wenn ein Schlüssel die Standard-Darstellung einer nicht-negativen integer Zahl ist, wird es als solche interpretiert werden, (z.B. `'8'` wird als `8` interpretiert während `'08'` als `'08'` interpretiert werden wird).

Ein Wert kann irgendetwas sein.

Weglassen von Schlüsseln. Falls Sie einen Schlüssel weglassen, wird das Maximum der Integer-Indizes gennnommen und der neue Schlüssel wird dieses Maximum + 1 sein. Falls es bis dahin keine Integer-Indizes gibt, wird der Schlüssel zu `0` (Null). Falls Sie einen Schlüssel angeben dem schon ein Wert zugeordnet wurde, wird dieser Wert überschrieben.

```
array( [Schlüssel =>] Wert
     , ...
     )
// Schlüssel ist entweder ein string oder ein nicht-negativer integer
// Wert kann irgendetwas sein.
```

Erzeugen / Verändern mit der eckigen Klammer-Syntax

Sie können ein bestehendes Array durch explizites Zuweisen von Werten verändern.

Weisen Sie dem Array Werte zu indem Sie den Schlüssel in eckigen Klammern angeben. Sie können den Schlüssel auch weglassen. In diesem Fall schreiben Sie einfach ein leeres Klammerpaar ("[]") hinter den Variablennamen.

```
$arr[Schlüssel] = Wert;
$arr[] = Wert;
// Schlüssel ist entweder ein string oder ein nicht-negativer integer
// Wert kann irgendetwas sein.
```

Falls `$arr` bis dahin nicht existiert, wird es erzeugt. Das ist also eine alternative Syntax um ein Array zu erzeugen. Um einen bestimmten Wert zu ändern, weisen Sie diesem einfach einen neuen Wert zu. Wenn Sie ein Schlüssel/Wert Paar entfernen möchten, benutzen Sie unset().

Nützliche Funktionen

Es gibt viele nützliche Funktionen um mit Arrays zu arbeiten. Schauen Sie unter dem Abschnitt Array Funktionen nach.

> **Anmerkung:** Mit unset() können Sie Schlüssel eines Arrays löschen. Halten Sie sich vor Augen, dass das Array NICHT neu indiziert wird.

```
$a = array( 1 => 'eins', 2 => 'zwei', 3 => 'drei' );
unset( $a[2] );

/* erzeugt ein Array das wie folgt definiert worden wäre:
   $a = array( 1=>'eins', 3=>'drei');
   und NICHT
   $a = array( 1 => 'eins', 2 => 'drei');
*/
```

Die foreach Kontrollstruktur gibt es speziell für Arrays. Damit ist es leicht ein Array zu durchlaufen.

Array do's und don'ts

Warum ist `$foo[bar]` falsch?

Vieleicht kennen Sie folgende Syntax aus alten Skripts:

```
$foo[bar] = 'Feind';
echo $foo[bar];
// usw
```

Es ist falsch, funktioniert aber. Warum ist es dann falsch? Der Grund ist folgender: Wie im Abschnitt syntax ausgeführt, muss zwischen den eckigen Klammern ('`[`' und '`]`')ein Ausdruck stehen. Das heißt, dass Sie Dinge ähnlich diesen schreiben können:

```
echo $arr[ foo(true) ];
```

Das ist ein Beispiel dafür, wie Sie den Rückgabewert einer Funktion als Array-Index verwenden können. Ebenso kennt PHP Konstanten, vielleicht haben Sie die `E_*` Konstanten schon einmal gesehen.

```
$error_descriptions[E_ERROR]   = "Ein schwerwiegender Fehler ist aufgetreten";
$error_descriptions[E_WARNING] = "PHP hat eine Warnung ausgegeben";
$error_descriptions[E_NOTICE]  = "Das ist nur eine informelle Notiz";
```

Beachten Sie, dass `E_ERROR` ebenso ein gültiger Bezeichner ist, genau wie `bar` im ersten Beispiel. Das letzte Beispiel entspricht aber tatsächlich dem folgendem Code:

```
$error_descriptions[1] = "Ein schwerwiegender Fehler ist aufgetreten";
$error_descriptions[2] = "PHP hat eine Warnung ausgegeben";
$error_descriptions[8] = "Das ist nur eine informelle Notiz";
```

weil `E_ERROR` gleich `1` ist, usw.

Aber wie ist es dann möglich dass `$foo[bar]` funktioniert? Es funktioniert, weil `bar` aufgrund seiner Syntax als Konstante ausgewertet wird. In diesem Fall jedoch existiert keine Konstane mit dem Namen `bar`. PHP nimmt an, dass Sie `bar` wörtlich gemeint haben, wie den String `'bar'`, aber vergessen haben, die Anführungszeichen zu setzen

Aber warum ist es schlecht?

Eines Tages könnte sich das PHP Team entscheiden eine weitere Konstante oder ein Schlüsselwort hinzuzufügen und dann geraten Sie in Schwierigkeiten. Zum Beispiel können Sie bereits die Wörter `empty` und `default` auf diese Art nicht benutzen, weil es speziele Schlüsselwörter sind.

Und, falls diese Argumente nicht ausreichen: diese Syntax ist schlicht veraltet und könnte irgendwann nicht mehr funktionieren.

> **Tipp:** Wenn Sie error_reporting auf `E_ALL` setzen, werden sie bemerken, dass PHP jedesmal eine Warnung ausgibt, wenn Sie dieses Konstrukt benutzen. Das gilt ebenso für andere veraltete 'Features'. (Fügen Sie die Zeile `error_reporting(E_ALL);` in Ihr Skript ein.)

Anmerkung: Innerhalb eines string in doppelten Anführunfszeichen gilt eine andere Syntax. Für mehr Details schauen Sie unter Variablen-Analyse in Strings nach.

Beispiele

Der Array-Typ in PHP ist sehr vielseitig. Hier folgen einige Beispiele um Ihnen die Mächtigkeit von Arrays zu veranschaulichen.

```
// Das hier
$a = array('Farbe'     => 'rot'
          ,'Geschmack' => 'süß'
          ,'Form'      => 'rund'
          ,'Name'      => 'Apfel'
          ,4                  // Schlüssel ist 0
          );

// ist komplett gleichwertig mit
$a['Farbe']     = 'rot';
$a['Geschmack'] = 'süß';
$a['Form']      = 'rund';
$a['Name']      = 'Apfel';
$a[]            = 4;          // Schlüssel ist 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// ergibt das Array: array( 0 => 'a' , 1 => 'b' , 2 => 'c' ),
// oder einfacher: array('a', 'b', 'c')
```

Beispiel 6-4. Verwenden von array()

```
// Array als (Eigenschafts-)Abbildung
$map = array('version'    => 4
            ,'OS'         => 'Linux'
            ,'Sprache'    => 'englisch'
            ,'short_tags' => true
            );

// strikt numerische Schlüssel
$array = array(7
              ,8
              ,0
              ,156
              ,-10
              );
// Dies entspricht: array( 0 => 7, 1 => 8, ...)
```

```
// wechselnde Schlüssel
$wechselnd = array(10          // Schlüssel: 0
                  ,5    =>  6
                  ,3    =>  7
                  ,'a'  =>  4
                  ,11          // Schlüssel: 6 (Maximum der Integer-Indices war 5)
                  ,'8'  =>  2  // Schlüssel: 8 (integer!)
                  ,'02' => 77  // Schlüssel: '02'
                  ,0    => 12  // 10 wird mit dem neuen Wert 12 überschrieben
                  );

// leeres Array
$leer = array();
```

Beispiel 6-5. Sammlung

```
$farben = array('rot','blau','grün','gelb');

foreach ($farben as $farbe) {
    echo "Gefällt Ihnen $farbe?\n";
}

/* Ausgabe:
Gefällt Ihnen rot?
Gefällt Ihnen blau?
Gefällt Ihnen grün?
Gefällt Ihnen gelb?
*/
```

Beachten Sie, dass es gegenwärtig nicht möglich ist die Werte eines Arrays direkt in einer solchen Schleife zu ändern. Eine Lösung ist folgendes:

Beispiel 6-6. Sammlung

```
foreach ($farben as $key => $farbe) {
    // funktioniert nicht:
    //$farbe = strtoupper($farbe);

    //funktioniert:
    $farben[$key] = strtoupper($farbe);
}
print_r($farben);

/* Ausgabe:
Array
(
    [0] => ROT
    [1] => BLAU
    [2] => GRÜN
```

```
    [3] => GELB
)
*/
```

Dieses Beispiel erzeugt ein Array, dessen Schlüssel bei eins beginnt.

Beispiel 6-7. 1-basierter Index

```
$erstes_viertel  = array(1 => 'Januar', 'Februar', 'März');
print_r($erstes_viertel);

/* Ausgabe:
Array
(
    [1] => 'Januar'
    [2] => 'Februar'
    [3] => 'März'
)
*/
```

Beispiel 6-8. Füllen eines realen Arrays

```
// Füllen eines Arrays mit allen Inhalten eines Verzeichnisses
$handle = opendir('.');
while ($file = readdir($handle))
{
    $files[] = $file;
}
closedir($handle);
```

Arrays sind geordnet. Diese Ordnung können Sie mit Hilfe verschiedener Sortierfunktionen verändern. Für mehr Informationen schauen Sie unter dem Abschnitt zu den Array Funktionen nach.

Beispiel 6-9. Sortien eines Array

```
sort($files);
print_r($files);
```

Weil der Wert eines Arrays irgendetwas sein kann, kann dieser Wert wiederum ein Array sein. Somit können Sie rekursive und multidimensionale Arrays erzeugen.

Beispiel 6-10. Rekursive and multidimensionale Arrays

```
$fruits = array ( "früchte"  => array("a" => "Orange"
                                      ,"b" => "Banane"
                                      ,"c" => "Apfel"
                                      )
                , "nummern"  => array(1
                                      ,2
                                      ,3
                                      ,4
                                      ,5
                                      ,6
                                      )
                , "löcher"   => array("erstens"
                                      ,5 => "zweitens"
                                      ,"drittens"
                                      )
                );
```

Objekte

Objekt-Initialisierung

Um ein Objekt zu initialisieren benutzen sie die Angabe `new`, dadurch wird das Objekt einer Variablen-Instanz zugewiesen.

```
<?php
class foo {
    function do_foo () {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo ();
?>
```

Für eine ausführliche Besprechung lesen Sie das Kapitel Klassen und Objekte

Resource

Eine Resource ist eine spezielle Variable, die eine Referenz auf eine externe Resource enthält. Resourcen werden von bestimmten Funktionen erzeugt und benutzt. Eine Liste dieser Funktionen und der entsprechenden Resource-Typen finden Sie im Anhang.

Anmerkung: Der Resource-Typ wurde in PHP 4 eingeführt.

Freigabe von Resourcen

Aufgrund des Reference-Counting Systems, das mit PHP 4's Zend-Egine eingeführt wurde, wird automatisch entdeckt, wenn auf eine Resource nicht mehr zugegriffen wird (wie in Java). Wenn dies der Fall ist werden alle Resourcen, die für diese Resource in Gebrauch waren, durch den "Müllsammler" (garbage collector) freigegeben. Aus diesem Grund ist es selten jemals notwendig, Speicher manuell durch Aufruf von free_result-Funktionen freizugeben.

Anmerkung: Persistente Datenbank-Links stellen einen Sonderfall dar, sie werden durch den gc (garbage collector) *nicht* zerstört. Siehe auch Persistente Datenbankverbindungen.

NULL

Der spezielle `NULL` Wert steht dafür, dass eine Varaiable keinen Wert hat. `NULL` ist der einzig mögliche Wert des Typs NULL.

Anmerkung: Der Null-Typ wurde in PHP 4 eingeführt.

Syntax

Es gibt nur einen Wert des Typs `NULL`, das Schlüsselwort `NULL`. Groß-/Kleinschreibung spielt keine Rolle.

```
$var = NULL;
```

Typen-Tricks

PHP erfordert (bzw. unterstützt) keine explizite Typ-Definitionen bei der Deklaration von Variablen; der Typ einer Variablen wird durch den Zusammenhang bestimmt in dem die Variable benutzt wird.

Das bedeutet, dass bei der Zuweisung einer Zeichenkette / eines Strings zu einer Variablen `var` diese Variable `var` zum Typ String wird. Sollten Sie dananch der Variablen `var` einen Integer-Wert zuweisen, wird sie zu einem Integer-Typ.

Ein Beispiel für die automatische Typ-Konvertierung von PHP ist der Plus-Operator '+'. Ist einer der zu addierenden Werte vom Typ float, werden alle Werte als float-Typ gbehandelt. Auch das Ergebnis der Addition wird vom Typ float sein. Andernfalls werden die Werte als integer-Typen angesehen und das Ergebnis wird ebenfalls vom Typ Integer sein. Beachten Sie, dass hierdurch NICHT der Typ der Operanden selbst beeinflusst wird; der Unterschied liegt einzig und allein in der Auswertung dieser Operanden.

```
$foo = "0";                          // $foo ist vom Typ String (ASCII 48)
$foo += 2;                           // $foo ist jetzt vom Typ Integer (2)
$foo = $foo + 1.3;                   // $foo ist nun vom Typ float (3.3)
$foo = 5 + "10 Sehr kleine Ferkel"; // $foo ist vom Typ Integer (15)
$foo = 5 + "10 Kleine Schweine";    // $foo ist vom Typ Integer (15)
```

Wenn Ihnen die beiden letzten Beispiele merkwürdig erscheinen, beachten Sie die String-Umwandlung.

Wenn Sie die Auswertung einer Variablen als einen speziellen Typ erzwingen wollen, beachten sie den Abschnitt in Typ-Umwandlung. Wollen Sie den Typ einer Variable ändern, schauen Sie bei settype() nach.

Um die Beispiele dieses Kapitels auszuprobieren, können Sie die Funktion var_dump() verwenden.

Anmerkung: Das Verhalten einer automatischen Umwandlung zum Typ Array ist zur Zeit nicht definiert.

```
$a = 1;       // $a ist ein Integer-Wert
$a[0] = "f";  // $a wird zum Array, $a[0] enthält "f"
```

Weil das o.a. Beispiel so aussieht, als ob $a ein Array wird, dessen erstes Element ein 'f' enthält, bedenken Sie folgendes:

```
$a = "1";     // $a ist ein String
$a[0] = "f";  // Was ist mit den String-Offsets? Was passiert?
```

Weil PHP die Indizierung innerhalb von Strings mittels der gleichen Syntax wie bei der Array-Indizierung unterstützt, führt das o.a. Beispiel zu einem Problem: Soll $a zu einem Array werden, dessen erstes Element ein "f" ist oder soll "f" das erste Zeichen des Strings $a werden?

Aus diesem Grund ist das Ergebnis der automatischen Umwandlung seit PHP 3.0.12 und auch in PHP 4.0b3-RC4 nicht definiert. Lösungen werden noch dikutiert.

Typ-Umwandlung

Typ-Umwandlung in PHP funktioniert vielfach wie in C: Der Name des gewünschten Typs wird vor der umzuwandelnden Variablen in Klammern gesetzt.

```
$foo = 10;           // $foo ist ein Integer-Wert
$bar = (float) $foo; // $bar ist vom Typ float
```

Folgende Umwandlungen sind möglich:

- (int), (integer) - nach integer
- (bool), (boolean) - nach boolean
- (float), (double), (real) - nach float
- (string) - nach string
- (array) - nach array
- (object) - Wandlung zum Objekt

Tipp: Ansttat eine Variable in einen String umzuwandeln, können Sie die Variable auch in doppelte Anführungszeichen einschließen.

Beachten sie, dass Tabulatoren und Leerzeichen innerhalb der Klammern erlaubt sind. Deshalb sind die folgenden Beispiele identisch:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Es ist nicht immer offenkundig, was bei der Typ-Umwandlung geschieht. Für weitere Informationen schauen Sie unter folgenden Abschnitten nach:

- Umwandlung nach boolean
- Umwandlung nach integer

Wenn Sie eine Umwandlung eines Arrays zu einem String vornehmen oder erzwingen, ist das Ergebnis das Wort `Array`. Wenn Sie eine Umwandlung eines Objekts zu einem String vornehmen oder erzwingen, ist das Ergebnis das Wort `Objekt`.

Bei der Umwandlung einer skalaren oder String-Variablen zu einem Array wird die Variable das erste Element des Arrays:

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];        // gibt 'ciao' aus
```

Sobald eine skalare oder String-Variable in ein Objekt gewandelt wird, wird die Variable zu einem Attribut des Objekts; der Eigenschafts-Name wird ’scalar’:

```
$var = ’ciao’;
$obj = (object) $var;
echo $obj->scalar;   // gibt ’ciao’ aus
```

Kapitel 7. Variablen

Grundlegendes

Variablen werden in PHP dargestellt durch ein Dollar-Zeichen ($) gefolgt vom Namen der Variablen. Bei Variablen-Namen wird zwischen Groß- und Kleinschreibung unterschieden (case-sensitive).

Variablen-Namen werden in PHP nach den gleichen Regeln wie andere Bezeichner erstellt. Ein gültiger Variablen-Name beginnt mit einem Buchstaben oder einem Unterstrich ("_"), gefolgt von einer beliebigen Anzahl von Buchstaben, Zahlen oder Unterstrichen. Als regulärer Ausdruck (regular expression) würde das wie folgt ausgedrückt: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'.

> **Anmerkung:** Unserem Zweck entspricht also ein Buchstabe von a bis z bzw. A bis Z oder einem ASCII-Zeichen von 127 bis 255 (0x7f bis 0xff).

```
$var = "Du";
$vaR = "und";
$Var = "ich";
$vAr = "wir lernen PHP"
echo "$var $vaR $Var, $vAr"; // gibt "Du und ich, wir lernen PHP" aus

$4site  = 'nicht jetzt';     // ungültig, da Anfang eine Zahl
$_4site = 'nicht jetzt';     // gültig, da Unterstrich am Anfang
$täbyte = 'irgendwas';       // gültig, da 'ä' dem ASCII-Wert 228 entspricht
```

Variablen werden in PHP 3 durch ihren Wert bestimmt. Das heisst, wenn sie einer Variablen einen Ausdruck zuweisen, wird der gesamte Inhalt des Originalausdrucks in die Zielvariable kopiert. Die Folge ist, dass eine Variable, die ihren Inhalt von einer anderen Variablen erhalten hat, ihren Inhalt behält, auch wenn sie danach den Inhalt der anderen (Quell- / Ursprungs-)Variablen ändern. Die Inhalte der Ziel- und Quellvariablen sind also insoweit unabhängig voneinander. Für weitere Informationen lesen sie bitte Expressions / Ausdrücke.

PHP 4 bietet eine andere Möglichkeit der Wertzuweisung bei Variablen: *Zuweisung durch Referenzierung*. Das bedeutet, dass der Wert der neuen Variablen eine Referenz zur Ursprungs-Variablen darstellt (mit anderen Worten: Der Wert ist ein Alias bzw. Zeiger auf den Inhalt der Ursprungsvariablen). Beide Variablen zeigen also auf die selbe(n) Speicherstelle(n). Änderungen der neuen Variablen ändern auch deren Ursprungs- Variable und umgekehrt. Der Wert / Inhalt wird also nicht kopiert. Die Übertragung geschieht dadurch auch schneller als in PHP 3. Dies wird sich aber nur bei umfangreichen Schleifen oder bei der Übertragung von grossen Arrays oder Objekten bemerkbar machen.

Für die Zuweisung per Referenz müssen sie lediglich ein & der (Ausgangs-, Quell-) Variablen voranstellen, die sie einer anderen Variablen zuweisen wollen. Der folgende Skript- Ausschnitt wird zweimal 'Mein Name ist Bob' ausgeben:

```
<?php
$foo = 'Bob';             // 'Bob' der Variablen $foo zuweisen.
$bar = &$foo;             // Zeiger auf $foo in $bar erzeugen.
$bar = "My name is $bar"; // $bar verändern...
echo $foo;                // $foo wurde dadurch ebenfalls verändert.
echo $bar;
```

```
?>
```

Zu beachten ist, dass nur Variablenbezeichner referenziert werden können.

```
<?php
$foo = 25;
$bar = &$foo;     // Gültige Zuweisung.
$bar = &(24 * 7); // Ungültig, da kein Variablenbezeichner
                  // zugewiesen wird.
function test() {
    return 25;
}

$bar = &();   // Ungültig.
?>
```

Vordefinierte Variablen

PHP bietet jedem ausgeführtem Skript eine Vielzahl von vordefinierten Variablen an. Viele dieser Variablen können jedoch nicht vollständig erläutert werden, da sie abhängig sind vom Web-Server, der Version und dem Setup des Web- Servers sowie weiteren Faktoren. Einige dieser Variablen stehen nicht zur Verfügung, wenn PHP-Skripte per Kommando- Zeilen-Aufruf ausgeführt werden.

Ungeachtet dieser Faktoren folgt nun eine Liste von vordefinierten Variablen innerhalb einer Basis-PHP 3- Installation als Modul und einer Basis-Installation des Apache (http://www.apache.org/)-Web-Servers in der Vers. 1.3.6.

Eine Liste aller vordefinierten Variablen (und weitere nützliche Informationen) erhalten sie durch Lesen der Infos zu (und den Gebrauch) der phpinfo()-Funktion.

> **Anmerkung:** Diese Liste ist weder vollständig noch erhebt sie Anspruch auf Vollständigkeit. Sie dient lediglich als Anleitung zu den möglicherweise zu erwartenden Arten von vordefinierten Variablen und dem Zugriff darauf.

Apache-Variablen

Diese Variablen werden durch den Apache (http://www.apache.org/)-Web-Server erzeugt. Sollten sie einen anderen Web-Server nutzen, gibt es keine Garantie, dass dieser die selben Variablen unterstützt. Es könnten einige sein; es könnten aber auch hier nicht aufgeführte sein. Viele dieser Variablen werden in den CGI 1.1 Spezifikationen (http://hoohoo.ncsa.uiuc.edu/cgi/env.html) aufgeführt. Darauf sollten sie vorbereitet sein.

Beachten Sie, dass nur wenige, wenn überhaupt, dieser Variablen zur Verfügung stehen (oder tatsächlich eine beliebige Bedeutung haben), wenn Sie PHP per Kommandozeile aufrufen.

$GATEWAY_INTERFACE

Die Revision der CGI-Spezifikation, die der Web-Server benutzt, z.B. 'CGI/1.1'.

$SERVER_NAME

Der Host-Name des Web-Servers, der das Skript ausführt. Das kann auch der Name eines virtuellen Servers sein.

$SERVER_SOFTWARE

Der Identifikations-String des Web-Servers (aus den Headern), sobald er Anforderungen beantwortet.

$SERVER_PROTOCOL

Name und Revision des Informations-Protokolls, über das die Seite angefordert wurde, z.B. 'HTTP/1.0'.

$REQUEST_METHOD

Welche Methode zum Zugriff auf die Seite benutzt wurde, z.B. 'GET', 'HEAD', 'POST', 'PUT'.

$QUERY_STRING

Der Abfrage-(Query-)String (falls vorhanden), mit dem auf die Seite zugegriffen wurde.

$DOCUMENT_ROOT

Das Verzeichnis des gerade ausgeführten Skripts aus Sicht des in der Konfigurations-Datei des Servers definierten Dokument-Wurzel-Verzeichnisses.

$HTTP_ACCEPT

Inhalt des `Accept:`-Headers der aktuellen Anforderung (so es einen solche Header gibt).

$HTTP_ACCEPT_CHARSET

Inhalt des `Accept-Charset:`-Headers der aktuellen Anforderung (sofern es einen gibt). Beispiel: 'iso-8859-1,*,utf-8'.

$HTTP_ENCODING

Inhalt des `Accept-Encoding:`-Headers der aktuellen Anforderung (wenn es einen gibt). Beispiel: 'gzip'.

$HTTP_ACCEPT_LANGUAGE

Inhalt des `Accept-Language:`-Headers in der aktuellen Anforderung, sofern dieser Header existiert. Beispiel: 'en'.

$HTTP_CONNECTION

Inhalt des `Connection:`-Headers des aktuellen Request, so er vorhanden ist. Beispiel: 'Keep-Alive'.

$HTTP_HOST

Inhalt des `Host:`-Headers der aktuellen Anforderung, wenn er existiert.

$HTTP_REFERER

Die Adresse (URL) - sofern vorhanden - der Seite, von der aus auf die aktuelle Seite gesprungen wurde. Dieser wird vom Browser des Benutzers gesetzt. Nicht alle Browser unterstützen dies.

$HTTP_USER_AGENT

Inhalt der `User_Agent:`-Header-Angabe der aktuellen Anfrage (wenn eine Angabe existiert). Dabei handelt es sich um eine Zeichenkette, welche der Browser benennt, mit dem die aktuelle Seite aufgerufen wurde, z.B. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586).` Abgesehen von anderen Dingen können sie diesen Wert zusammen mit der get_browser() Funktion dazu verwenden, ihre Web-Seiten in Abhängigkeit von den Möglichkeiten des jeweils benutzten Browsers anzeigen zu lassen.

$REMOTE_ADDR

Die IP-Adresse, von der aus gerade auf die Web-Seite zugegriffen wird.

$REMOTE_PORT

Der Port, der zum Zugriff auf ihren Web-Server seitens des Anwender-Computers benutzt wird.

$SCRIPT_FILENAME

Der absolute Pfadname des gerade ausgeführten Skripts.

$SERVER_ADMIN

Der Inhalt der in der Konfigurations-Datei des Web-Servers (Apache) stehenden Angabe zum SERVER_ADMIN. Bei virtuellen Hosts der dafür eingetragene Wert.

$SERVER_PORT

Der vom Server für den Web-Server benutzte Kommunikations- Port (normalerweise ’80’). Verwenden sie z.B. SSL, wird dieser Port derjenige sein, den sie für sicheres HTTP definiert haben.

$SERVER_SIGNATURE

Sofern eingeschaltet, enthält diese Zeichenkette die Server- Version und den virtuellen Host-Namen. Sie wird den durch den Server generierten Seiten hinzu gefügt.

$PATH_TRANSLATED

In Abhängigkeit vom Datei- / File-System der Pfad des aktuellen Skripts, nachdem der Server das virtuelle Mapping in ein reales Mapping umgesetzt hat (nicht der Dokument-Wurzel-Pfad).

$SCRIPT_NAME

Enthält den Pfad des aktuellen Skripts. Nützlich für Seiten, die auf sich selbst verweisen müssen.

$REQUEST_URI

Die URI, die durch den Zugriff auf die aktuelle Seite gegeben ist, z.B. ’/index.html’.

Umgebungs- / Environment-Variablen

Diese Variablen werden aus der Umgebung, in der PHP läuft, in den globalen Namensbereich von PHP importiert. Viele werden durch die jeweilige Shell, in der PHP läuft, unterstützt bzw. gebildet. Da es verschiedenste Systemumgebungen mit den unterschiedlichsten Shell‘s gibt, ist es nicht möglich, eine abschließende Liste der definierten Umgebungs-Variablen aufzustellen. Lesen sie deshalb in der Anleitung zu ihrer Shell nach, um eine Liste dieser systembezogenen Variablen zu erhalten.

Andere Umgebungs-Variablen beinhalten die CGI-Variablen, die ohne Rücksicht darauf, ob PHP als Web-Server-Modul oder im CGI-Modus läuft, gesetzt werden.

PHP-Variablen

Diese Variablen werden durch PHP selbst erzeugt. `$HTTP_*_VARS` Variablen stehen nur zur Verfügung, wenn die Option track_vars in der php.ini auf "on" gesetzt ist. Wenn dies der Fall ist, werden diese Variablen immer gesetzt, selbst wenn es leere Arrays sind. Das verhindert, dass ein böswilliger Nutzer diese Variablen manipuliert.

> **Anmerkung:** Seit PHP 4.0.3 ist track_vars immer aktiviert, ohne Rücksicht auf die Einstellungen in der Konfigurationsdatei.

> **Anmerkung:** Die neuen "Superglobals" stehen seit der PHP Version 4.1.0. zur Verfügung. Im 4.1.0 Release Announcement (http://www.php.net/release_4_1_0.php) können Sie mehr Details nachlesen. Dieses sind die Arrays `$_GET`, `$_POST`, `$_ENV`, `$_SERVER`, `$_COOKIE`, `$_REQUEST` `$_FILES` und `$_SESSION` und werden informell als *Superglobals* bezeichnet, weil sie immer zur Verfügung stehen, ohne Berücksichtigung des Geltungsbereichs.Damit sind die alten, beziehungsweise die `$HTTP_*_VARS` Arrays veraltet.

Wenn register_globals aktiviert ist,stehen auch diese Variablen im globalen Namensbereich des Skripts zur Verfügung; z.B. getrennt von den Arrays `$HTTP_*_VARS` und `$_*`. Verwandte Informationen erhalten Sie im Kapitel über Sicherheit unter dem Abschnitt Verwendung von Register Globals.

$argv

Ein Array von Argumenten, die dem Skript übergeben werden. Wird das Skript an der Befehlszeile aufgerufen, ermöglicht dies C-ähnlichen Zugriff auf die Kommando-Zeilen-Parameter. Beim Aufruf per GET-Methode enthält dieses Array die Abfragewerte.

$argc

Anzahl der per Kommando-Zeile dem Skript übergebenen Parameter (wenn von dort aufgerufen).

$PHP_SELF

Der Dateiname des gerade ausgeführten Skripts, relativ zum Wurzel-Verzeichnis des Dokuments. Bei Kommando-Zeilen- Aufrufen ist diese Variable nicht verfügbar.

$HTTP_COOKIE_VARS

Ein assoziatives Array von Variablen, das dem aktuellen Skript über HTTP-Cookies übergeben wurde.

$_COOKIE

Ein assoziatives Array von Variablen, das dem aktuellen Skript über HTTP-Cookies übergeben wurde. Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

$HTTP_GET_VARS

Ein assoziatives Array von Variablen, das dem aktuellen Skript per HTTP-GET-Methode übergeben wurde.

$_GET

Ein assoziatives Array von Variablen, das dem aktuellen Skript per HTTP-GET-Methode übergeben wurde.Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

$HTTP_POST_VARS

Ein assoziatives Array aus Varaiblen, welches dem aktuellen Skript per HTTP-POST-Methode übergeben wurde.

$_POST

Ein assoziatives Array aus Variablen, welches dem aktuellen Skript per HTTP-POST-Methode übergeben wurde. Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

$HTTP_POST_FILES

Ein assoziatives Array aus Variablen, das Informationen über per HTTP POST-Methode hochgeladene Dateien enthält. Siehe Dateiuploads mit POST für Informationen über den Inhalt der `$HTTP_POST_FILES`. Eingeführt in PHP 4.0.0.

$_FILES

Ein assoziatives Array aus Variablen, das Informationen über per HTTP POST-Methode hochgeladene Dateien enthält. Siehe Dateiuploads mit POST für Informationen über den Inhalt der `$_FILES`.Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

$HTTP_ENV_VARS

Ein assoziatives Array aus Variablen, die dem aktuellen Skript über die Umgebung zur Verfügung stehen.

$_ENV

Ein assoziatives Array aus Variablen, die dem aktuellen Skript über die Umgebung zur Verfügung stehen. Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

$HTTP_SERVER_VARS

Ein assoziatives Array aus Variablen, die dem aktuellen Skript vom jeweiligen HTTP-Server übermittelt werden. Diese Variablen sind analog zu den oben beschriebenen Apache-Variablen.

$_SERVER

Ein assoziatives Array aus Variablen, die dem aktuellen Skript vom jeweiligen HTTP-Server übermittelt werden. Diese Variablen sind analog zu den oben beschriebenen Apache-Variablen.

Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

$HTTP_SESSION_VARS

Ein assoziatives Array aus Session-Variablen, die dem aktuellen Skript übergeben wurden.

$_SESSION

Ein assoziatives Array aus Session-Variablen, die dem aktuellen Skript übergeben wurden. Automatisch global in jedem Geltungsbereich. Werden dem Array $_SESSION neue Einträge hinzugefügt, werden diese automatisch als Session-Variablen registriert, genau so als ob die Funktion session_register() aufgerufen worden wäre. Eingeführt in PHP 4.1.0.

$_REQUEST

Ein assoziatives Array zusammengesetzt aus den GET, POST und Cookie Variablen. Mit anderen Worten - alle Informationen die vom Nutzer kommen und denen aus Sichtweise der Sicherheit nicht zu trauen ist. Automatisch global in jedem Geltungsbereich. Eingeführt in PHP 4.1.0.

Geltungsbereich von Variablen

Der Geltungsbereich einer Variablen ergibt sich aus dem Zusammenhang, in dem sie definiert wurde. Meistens besteht dieser aus einem einzigen Bereich. Dieser beinhaltet auch den Bereich für Dateien, die per "include"- oder "require"-Anweisung eingebunden wurden, z.B.:

```
$a = 1;
include "b.inc";
```

Die Variable $a ist auch in der eingebundenen Datei `b.inc` verfügbar. In benutzerdefinierten Funktionen wird ein auf die Funktion beschränkter Geltungsbereich eingeführt. Jede in einer Funktion benutzte Variable ist zunächst auf den lokalen Bereich der Funktion beschränkt, z.B.:

```
$a = 1; // globaler Bereich

function test () {
    echo $a; // Referenz auf einen lokalen Variablen-Bereich
}

test ();
```

Dieses Skript erzeugt keine Bildschirm-Ausgabe, da sich die Echo- Anweisung auf eine lokale Variable namens `$a` bezieht und dieser kein Wert im lokalen Bezug zugewiesen worden ist. Dies ist ein kleiner Unterschied zu C, wo globale Variablen auch in Funktionen vorhanden sind, es sei denn, sie werden durch eine funktionsinterne Definition überschrieben. Das kann zu Problemen führen,

denn in PHP müssen global geltende Variablen innerhalb von Funktionen als solche definiert werden. Ein Beispiel:

```
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
```

Das obige Skript gibt "3" aus. Durch das Deklararieren der Variablen `$a` und `$b`innerhalb der Funktion als global, weisen alle Referenzen zu beiden Variablen auf die nun globalen Werte. Es gibt keine Beschränkungen bei der Anzahl an globalen Variablen, die durch eine Funktion verändert werden können.

Eine weitere Möglichkeit besteht in der Verwendung des speziellen `$GLOBALS` PHP-Array. Das obige Beispiel kann damit auch so geschrieben werden:

```
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum();
echo $b;
```

Das `$GLOBALS`-Array ist ein assoziatives Array mit dem Bezeichner der globalen Variablen als Schlüssel und dem Inhalt dieser Variablen als Wert des Array-Elements.

Ein weiterer wichtiger Anwendungszweck von Variablen-Bereichen ist die *static*-Variable. Eine statische Variable existiert nur in einem lokalen Funktions-Bereich, der Wert geht beim Verlassen dieses Bereichs aber nicht verloren. Schauen sie das folgende Beispiel an:

```
function test ()
{
    $a = 0;
    echo $a;
    $a++;
}
```

Diese Funktion ist sinnlos, da sie bei jedem Aufruf `$a` auf `0` setzt und "0" ausgibt. Die Anweisung `$a++`, welche den Wert erhöht, macht keinen Sinn, da der Wert von `$a` beim Verlassen der Funktion verloren geht. Um eine sinnvolle Zählfunktion zu implementieren, die ihren aktuell gesetzten Wert nicht vergisst, müssen sie die Variable `$a`als "static" deklarieren:

```
function test()
{
    static $a = 0;
    echo $a;
    $a++;
}
```

Jetzt wird bei jedem Aufruf der test()-Funktion der aktuelle Wert von `$a` ausgegeben und dann um 1 erhöht.

Static-Variablen ermöglichen auch einen Weg zum Umgang mit rekursiven Funktionen. Das sind Funktionen, die sich selbst aufrufen. Hierbei besteht die Gefahr, so genannte Endlos- Schleifen zu programmieren. Sie müssen also einen Weg vorsehen, diese Rekursion zu beenden. Die folgende einfache Funktion zählt rekursiv bis 10. Die statische Variable `$count` wird benutzt, um die Rekursion zu beenden:

```
function test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        test ();
    }
    $count--;
}
```

Variable Variablen

Manchmal ist es komfortabel, variable Variablen-Bezeichner zu benutzen. Das bedeutet, einen Variablen-Namen zu setzen und dynamisch zu gebrauchen. Eine normale Variable wird wie folgt gebildet:

```
$a = "Hallo";
```

Eine variable Variable nimmt den Wert einer Variablen und behandelt ihn als Bezeichner der Variablen. Im obigen Beispiel kann *Hallo* als Variablen-Name gebraucht werden, indem man zwei $-Zeichen benutzt, also schreibt:

```
$$a = "Welt";
```

Nun existieren in der PHP-Symbol-Struktur zwei definierte und gespeicherte Variablen: `$a` mit dem Inhalt "Hallo" und `$Hallo` mit dem Inhalt "Welt". Deshalb wird die Anweisung

```
echo "$a ${$a}";
```

zur genau gleichen Ausgabe führen wie:

```
echo "$a $Hallo";
```

also zu: `Hallo Welt`.

Wenn sie variable Variablen mit Arrays verwenden, müssen sie eine Doppeldeutigkeit beachten. Wenn sie nämlich `$$a[1]` schreiben, dann muss der Parser wissen, ob Sie `$a[1]` als Variable oder `$$a` als Variable und dann [1] als Index dieser Variablen verwenden wollen bzw. gemeint haben. Die Syntax zur Lösung dieser Doppeldeutigkeit: Verwenden Sie im ersten Fall `${$a[1]}` und im zweiten Fall `${$a}[1]`.

Variablen ausserhalb von PHP

HTML-Formulare (GET and POST)

Sobald ein Formular an ein PHP-Skript übergeben wird, wird jede Variable dieses Formulars dem Skript automatisch verfügbar gemacht. Dafür sorgt PHP. Falls die Konfigurationsoption track_vars auf "on" gesetzt ist, werden diese Variablen in diesen assoziativen Arrays abgelegt `$HTTP_POST_VARS`, `$HTTP_GET_VARS` und/oderr `$HTTP_POST_FILES`, entsprechend der Quelle, aus der die fragliche Variable kommt.

Für weitere Informationen über diese Variablen lesen Sie bitte den Abschnitt über Vordefinierte Variablen.

Beispiel 7-1. Einfache Formular-Variablen

```
<form action="foo.php" method="post">
    Name: <input type="text" name="username"><br>
    <input type="submit">
</form>
```

Wird dieses Formular abgeschickt, steht der Wert des Textfeldes in der Variable `$HTTP_POST_VARS['username']` zur Verfügung. Wenn in der Konfigurationsdatei die Option register_globals auf "on" gesetzt ist, steht die Variable auch als `$username` global zur Verfügung.

> **Anmerkung:** Die Konfigurationseinstellung zu magic_quotes_gpc betrifft Get, Post and Cookie Werte. Ist diese Einstellung aktiv wird der Wert (It's "PHP!") automatisch zu (It\'s \"PHP!\"). Escaping ist notwendig, wenn Sie ihre Daten in eine Datenbank einfügen wollen. Siehe auch: addslashes(), stripslashes() und magic_quotes_sybase.

Im Zusammenhang mit Formular-Variablen versteht PHP auch Arrays (siehe auch die verwandte Faq). Sie können z.B. die betreffenden Variablen gruppieren oder dieses Leistungsmerkmal nutzen, um Werte aus Mehrfach-Auswahl-Bereichen zu erhalten.

Beispiel 7-2. Komplexere Formular-Variablen

```
<form action="array.php" method="post">
    Name:  <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Bier: <br>
    <select multiple name="bier[]">
        <option value="binding">Binding
        <option value="warsteiner">Warsteiner
        <option value="stauder">Stauder
        <option value="stuttgarter">Stuttgarter Schwabenbräu
    </select>
    <input type="submit">
</form>
```

In PHP 3 ist die Verwendung von Arrays in Formularvariablen auf eindimensionale Arrays beschränkt. In PHP 4 besteht diese Einschränkung nicht mehr.

IMAGE SUBMIT Variablen-Bezeichner

Zur Übertragung eines Formulars kann auch ein Bild (Image) statt eines Übertragungs-Schalters (Submit-Button) benutzt werden, dessen Tag wie folgt aussieht:

```
<input type="image" src="image.gif" name="sub">
```

Klickt der Benutzer irgendwo auf das Bild, wird das entsprechende Formular an den Web-Server übertragen. Hierbei sind zwei zusätzliche Variablen vorhanden, sub_x und sub_y. Diese enthalten die Koordinaten des Klick-Punktes innerhalb des Bildes. Die Erfahreneren werden sagen, dass die

Variablen, die vom Browser gesendet werden einen Punkt enthalten statt eines Unterstrichs. Dieser Punkt wird von PHP automatisch in einen Unterstrich verwandelt.

HTTP-Cookies

PHP unterstützt HTTP-Cookies, wie sie in Netscape's Spec (http://www.netscape.com/newsref/std/cookie_spec.html) definiert sind. Cookies ermöglichen die Daten-Speicherung innerhalb der jeweiligen Browser-Umgebung zur Weiterleitung oder wiederholten Identifikation von Benutzern. Sie können Cookies erzeugen, indem sie die Funktion setcookie() benutzen. Cookies sind Teil des HTTP-Headers, deshalb muss die setcookie-Funktion aufgerufen werden, bevor irgendeine Ausgabe an den Browser gesendet wird. Dabei handelt es sich um die gleiche Einschränkung, die auch für die header()-Funktion gilt. Alle Cookies, die der Client an sie sendet werden in PHP-Variablen umgesetzt, genau wie die Daten bei den GET- und POST-Methoden.

Wenn sie einem einzelnen Cookie mehrere Wert zuweisen wollen müssen sie dem Cookie-Namen *[]* hinzufügen. Z.B.:

```
SetCookie ("MeinCookie[]", "Ich teste", time()+3600);
```

Bedenken sie, dass ein Cookie ein vorhergehendes Cookie gleichen Namens überschreibt, es sei denn, der Pfad oder die Domain ist anders. Für eine Warenkorb-Anwendung können Sie deshalb z.B. einen Zähler bilden und diesen weiterleiten:

Beispiel 7-3. SetCookie-Beispiel

```
$Count++;
setcookie("Count", $Count, time()+3600);
setcookie("Cart[$Count]", $item, time()+3600);
```

Umgebungs- / Environment-Variablen

PHP sorgt automatisch für die Verfügbarkeit der Umgebungs- Variablen als normale PHP-Variablen.

```
echo $HOME;  /* Zeigt die HOME-Umgebungs-Variable (sofern gesetzt). */
```

Da Informationen per GET, POST und Cookie-Mechanismen übergeben werden, ist es manchmal das Beste, Umgebungs-Variablen explizit auszulesen. Dadurch wird die richtige Version eingelesen.

Hierfür kann die getenv()-Funktion genutzt werden. Den Wert einer Umgebungs-Variablen können Sie per putenv()- Funktion setzen.

Punkte in eingelesenen Variablen-Bezeichnern

Normalerweise verändert PHP die Variablen-Bezeichner nicht, wenn sie einem Skript übergeben werden. Es sollte aber beachtet werden, dass der Punkt (".") kein gültiger Bestandteil eines Variablen-Bezeichners ist. Deshalb achten sie auf folgendes:

```
$varname.ext;  /* ungültiger Variablen-Bezeichner */
```

Der PHP-Parser sieht eine Variable namens `$varname`, gefolgt von einem Zeichenketten-Verbindungs-Operator, dieser wiederrum gefolgt von der offenen Zeichenkette 'ext' (also nicht eingegrenzt durch '"' und auch keinem Schlüssel oder reserviertem Bezeichner entsprechend). Das kann natürlich nicht zum gewünschten Ergebnis führen.

Deshalb ist es wichtig zu wissen, dass PHP in den ihm übergebenen Variablen alle Punkte (.) automatisch durch einen Unterstrich (_) ersetzt.

Bestimmung des Variablen-Typs

Da PHP den Typ der Variablen bestimmt und (im Allgemeinen) selbst eine entsprechende Umformung vornimmt, ist es nicht immer klar, welchen Typ eine Variable gerade hat. PHP beinhaltet einige Funktionen, die dies herausfinden. Als da sind: gettype(), is_long(), is_double(), is_string(), is_array() und is_object().

Kapitel 8. Konstanten

PHP definiert eine Reihe von Konstanten und stellt einen Mechanismus zur Verfügung, mit dem man zusätzliche Konstanten zur Laufzeit definieren kann. Konstanten sind Variablen sehr ähnlich, bis auf die Tatsache, dass sie mit der define()-Funktion definiert werden müssen und später nicht mehr mit einem anderen Wert versehen werden können.

Folgende vordefinierten Variablen sind immer verfügbar:

__FILE__

Der Name der Skript-Datei, die gerade geparsed wird. Wird diese Konstante in einer Datei verwendet, die per include() oder require() eingebunden wurde, liefert sie den Namen der eingebundenen Datei, nicht den der aufrufenden Datei.

__LINE__

Die Nummer der Zeile im laufenden Skript, die gerade geparst wird. Wird diese Konstante in einer Datei benutzt, die per include() oder require() eingebunden wurde, liefert sie die Zeilennummer innerhalb der eingebundenen Datei.

PHP_VERSION

Ein String, der die Versionsnummer des PHP-Parsers enthält, der gerade verwendet wird; z. B.´3.0.8-dev´.

PHP_OS

Der Name des Betriebsystems, auf dem der PHP-Parser ausgeführt wird; z. B. ´Linux´.

TRUE

Der Wert ´wahr´.

FALSE

Der Wert ´falsch´.

E_ERROR

Bedeutet einen Fehler, der sich von einem ´parsing error´ unterscheidet. Die Ausführung des Skriptes wird beendet.

E_WARNING

Gibt einen Zustand zurück,durch den PHP weiß, dass etwas nicht in Ordnung ist, das aktuelle Skript aber trotzdem weiter ausführt; dies kann vom Skript selbst aufgefangen werden. Ein Beispiel wäre ein ungültiger regulärer Ausdruck (regexp) in der Funktion ereg().

E_PARSE

Der Parser hat Probleme mit ungültiger Syntax in der Skript-Datei. Die Ausführung des Skriptes wird beendet.

E_NOTICE

Etwas ist aufgetreten, das ein Fehler sein kann oder nicht. Das aktuelle Skript wird weiter ausgeführt. Beispiele hierfür sind ein nicht gequoteter string als Hash-Index oder der Zugriff auf eine Variable, die nicht gesetzt wurde.

Die E_*-Konstanten werden typischerweise mit der error_reporting()-Funktion benutzt, um das Fehlermeldungs-Niveau festzusetzen.

Zusätzliche Konstanten können mithilfe der define()-Funktion definiert werden.

Zu beachten ist, dass dies Konstanten sind, und keine Makros, wie man sie von C her kennt; nur gültige Skalar-Daten können von einer Konstante vertreten werden.

Beispiel 8-1. Konstanten definieren

```
<?php
define("CONSTANT", "Hallo Welt.");
echo CONSTANT; // gibt  "Hallo Welt." aus.
?>
```

Beispiel 8-2. Die Benutzung von __FILE__ und __LINE__

```
<?php
function report_error($file, $line, $message) {
    echo "Ein Fehler ist aufgetreten in $file in Zeile $line: $message.";
}

report_error(__FILE__,__LINE__, "Irgendetwas ist falsch gelaufen!");
?>
```

Kapitel 9. Ausdrücke

Ausdrücke (Expressions) sind die wichtigsten Bausteine von PHP. In PHP ist fast alles, was geschrieben wird, ein Ausdruck. Die einfachste, aber auch zutreffenste Definition für einen Ausdruck ist "alles, was einen Wert hat".

Die grundlegendste Formen von Ausdrücken sind Konstanten und Variablen. Wenn man "$a = 5" schreibt, weist man $a den Ausdruck ´5´ zu. ´5´ hat offensichtlich den Wert 5. Anders ausgedrückt: ´5´ ist ein Ausdruck mit dem Wert 5 (in diesem Fall ist ´5´ eine Integer-Konstante).

Nach dieser Zuweisung würde man erwarten, dass der Wert von $a nun ebenfalls 5 ist, wenn man also $b = $a schreibt, sollte dasselbe dabei herauskommen, als hätte man $b = 5 geschrieben. Anders ausgedrückt: $a wäre ebenfalls ein Ausdruck mit dem Wert 5. Wenn alles richtig funktioniert, wird genau das passieren.

Etwas kompliziertere Beispiele für Ausdrücke sind Funktionen:

```
function foo () {
    return 5;
}
```

Angenommen, Sie sind mit dem Konzept von Funktionen vertraut (wenn Sie es nicht sind, lesen Sie das Kapitel über Funktionen), dann würden Sie annehmen, dass die Eingabe von `$c = foo()` grundsätzlich daselbe bedeutet, als würde man schreiben `$c = 5`,und genau das trifft zu. Funktionen sind Ausdrücke mit dem Wert ihres Rückgabewertes. Da foo() den Wert 5 zurückgibt, ist der Wert des Ausdruckes ´foo()´ 5. Normalerweise geben Funktionen nicht einfach einen statischen Wert zurück, sondern berechnen irgendetwas.

Natürlich müssen Werte in PHP keine Integer-Zahlen sein, und oft sind sie es auch nicht. PHP unterstützt drei skalare Datentypen: integer values (Integer-Zahlen),floating point values (Fließkommazahlen) und string values (Zeichenketten). (Skalare sind Datentypen, die man nicht in kleinere Stücke ´brechen´ kann, im Gegensatz zu Arrays). PHP unsterstützt auch zwei zusammengesetzte (nicht-skalare) Datentypen: Arrays und Objekte. Jeder dieser Datentypen kann Variablen zugewiesen und von Funktionen zurückgegeben werden.

Bis hierhin sollten Benutzer von PHP/FI 2 keine Veränderung bemerkt haben. PHP fasst den Begriff ´Ausdruck´ aber noch viel weiter, wie es auch andere Programmiersprachen tun. PHP ist in dem Sinne eine ausdrucksoriente Sprache, dass fast alles ein Ausdruck ist. Zurück zu dem Beispiel, mit dem wir uns schon beschäftigt haben: ´$a = 5´. Es ist einfach zu erkennen, dass hier zwei Werte enthalten sind: Der Wert der Integer-Konstanten ´5´ und der Wert von $a, der auf 5 geändert wird. In Wirklichkeit ist aber noch ein zusätzlicher Wert enthalten, nämlich der Wert der Zuweisung selbst. Die Zuweisung selbst enthält den zugewiesenen Wert, in diesem Fall 5. In der Praxis bedeutet dies, dass ´$a = 5´, egal was es tut, immer einen Ausdruck mit dem Wert 5 darstellt. Folglich ist ´$b = ($a = 5)´ gleichbedeutend mit ´$a = 5; $b = 5;´ (ein Semikolon markiert das Ende einer Anweisung). Da Wertzuweisungen von rechts nach links geparst werden, kann man auch ´$b = $a = 5´ schreiben.

Ein anderes gutes Beispiel für die Ausdrucksorientierung von PHP sind Prä- und Post-Inkrement sowie die entsprechenden Dekremente. Benutzer von PHP/FI 2 und vielen anderen Sprachen sind vermutlich mit den Notationen ´variable++´ und ´variable--´ vertraut. Dies sind Inkrement- und Dekrement-Operatoren. IN PHP/FI 2 hat das Statement ´$a++´ keinen Wert (es ist kein Ausdruck) und daher kann man es nicht als Wert zuweisen oder in irgendeiner Weise benutzen. PHP erweitert die Eigenschaften von Dekrement und Inkrement, indem es die beiden ebenfalls zu Ausdrücken macht. In PHP gibt es, wie in C, zwei Arten von Inkrementen - Prä-Inkrement und Post-Inkrement. Grundsätzlich erhöhen sowohl Prä- als auch Post-Inkrement den Wert der Variable, und der Effekt auf die Variable ist derselbe. Der Unterschied ist der Wert des Inkrement-Ausdruckes. Das

Prä-Inkrement, das ´++$variable´ geschrieben wird, enthält als Wert den Wert der erhöhten Variabe (PHP erhöht den Wert der Variable, bevor es ihren Wert ausliest, daher der Name ´PRÄ-Inkrement´). Das Post-Inkrement, dass ´$variable++´ geschrieben wird, enthält dagegen den ursprünglichen Wert der Variablen vor der Erhöhung (PHP erhöht den Wert der Variablen, nachdem es ihren Wert ausgelesen hat, daher der Name ´POST-Inkrement´).

Ein sehr gebräuchlicher Typ von Ausdrücken sind Vergleichsausdrücke. Diese Ausdrücke geben entweder 0 (=FALSCH) oder 1 (=WAHR) zurück. PHP unterstützt > (größer), >= (größer oder gleich), == (gleich), != (ungleich), > (kleiner), und >= (kleiner oder gleich). Diese Ausdrücke werden meistens in bedingten Anweisungen, wie z. B. in `if`-Anweisungen, verwendet.

Im letzten Beispiel für Ausdrücke befassen wir uns mit kombinierten Zuweisungs- und Operator-Ausdrücken. Wir haben schon gezeigt, wie man den Wert von $a um 1 erhöht, nämlich indem man einfach ´$a++´ oder ´++$a´ schreibt. Aber was tut man, wenn man den Wert um mehr als eins erhöhen will, z. B. um 3? Man könnte mehrer Male ´$a++´ schreiben, aber das ist offensichtlich weder effizient noch sehr komfortabel. Viel üblicher ist es, einfach ´$a = $a + 3´ zu schreiben. ´$a + 3´ gibt den Wert von $a plus 3 zurück, dieser wird wieder $a zugewiesen, was dazu führt, dass $a nun um 3 erhöht wurde. In PHP - wie in einigen anderen Programmiersprachen, z. B. in C - kann man dies aber noch kürzer schreiben, was mit der Zeit klarer wird und auch einfacher zu verstehen ist. Um 3 zu dem aktuellen Wert hinzuzufügen, schreibt man ´$a += 3´. Das bedeutet exakt: "Nimm´ den Wert von $a, addiere 3 hinzu und weise $a den entstandenen Wert zu". Zusätzlich dazu, dass diese Schreibweise kürzer und klarer ist, resultiert sie auch in einer schnelleren Ausführung. Der Wert von ´$a += 3´ ist, wie der Wert einer regulären Zuweisung, der zugewiesene Wert. Es ist zu beachten, dass dieser Wert NICHT 3, sondern dem kombinierten Wert von $a plus 3 entspricht (Das ist der Wert, der $a zugewiesen wird). Jeder Operator, der zwei Elemente verbindet, kann in dieser Schreibweise verwendet werden, z. B. ´$a -= 5´ (vermindert den Wert von $a um 5) oder ´$a *= 7´ (multipliziert den Wert von $a mit 7 und weist das Ergebnis $a zu), usw.

Es gibt einen weiteren Ausdruck, der Ihnen vielleicht seltsam vorkommt, wenn Sie ihn bisher noch in keiner Programmiersprache kennengelernt haben, den dreifach konditionalen Operator:

```
$eins ? $zwei : $drei
```

Wenn der Wert des ersten Sub-Ausdruckes (hier: $eins) wahr ist (d. h. nicht `NULL`), dann wird der Wert des zweiten Subausdrucks (hier: $zwei) zurückgeben und ist das Ergebnis des konditionalen Ausdrucks. Andernfalls (d. h. wenn der erste Ausdruck falsch ist), wird der Wert des dritten Subausdruckes (hier: $drei) zurückgeben.

Das folgende Beispiel sollte das Verständnis von Prä- und Post-Inkrement und von Ausdrücken im allgemeinen erleichtern:

```
function verdoppeln($i) {
    return $i*2;
}
$b = $a = 5;        /* Weise den Variablen $a und $b beiden den Wert 5 zu */
$c = $a++;          /* Post-Inkrement, der urspruengliche Wert von $a (also 5)
                       wird $c zugewiesen. */
$e = $d = ++$b;     /* Prae-Inkrement, der erhöhte Wert von $b (= 6) wird $d und
                       $e zugewiesen. */

/* An diesem Punkt sind $d und $e beide gleich 6 */

$f = verdoppeln($d++);  /* Weise $f den doppelten Wert von $d vor
                       der Erhöhung um eins zu, 2*6 = 12 */
$g = double(++$e);  /* Weise $g den doppelten Wert von $e nach
                       der Erhoehung zu, 2*7 = 14 to $g */
```

```
$h = $g += 10;      /* Zuerst wie $g um 10 erhöht und hat schliesslich den Wert
                       24. Der Wert dieser Zuweisung (24) wird dann $h zugewiesen,
                       womit $h ebenfalls den Wert von 24 hat. */
```

Am Anfang dieses Kapitels hatten wir gesagt, wir würden die verschiedenen Arten von Anweisungen beschreiben und, wie versprochen, Ausdrücke können Anweisungen sein. Trotzdem ist nicht jeder Ausdruck eine Anweisung. In diesem Fall hat eine Anweisung die Form ´Ausdr´ ´;´, d. h. ein Ausdruck gefolgt von einem Semikolon. In ´$b=$a=5;´ ist $a=5 ein gültiger Ausdruck, aber für sich allein keine Anweisung. ´$b=$a=5;´ ist jedoch eine gültige Anweisung.

Ein letzter Punkt, der noch zu erwähnen ist, ist der ´wahr´-Wert von Ausdrücken. In vielen Fällen, hauptsächlich in bedingten Anweisungen und Schleifen, ist man nicht am spezifischen Wert eines Ausdrucks interessiert, sondern kümmert sich nur darum, ob er WAHR oder FALSCH bedeutet (PHP hat keinen speziellen boolean-Datentyp). Der Wahrheitswert eines Ausdrucks in in PHP wird ähnlich bestimmt wie in Perl. Jeder numerische Wert, der nicht `NULL` ist, bedeutet WAHR, `NULL` bedeutet FALSCH. Es ist zu beachten, dass negative Werte nicht `NULL` sind und deshalb als WAHR aufgefasst werden! Eine leere Zeichenkette und die Zeichenkette "0" sind FALSCH; alle anderen Zeichenketten sind WAHR. Nicht-skalare Datentypen (Arrays und Objekte) werden als FALSCH betrachtet, wenn sie keine Elemente enthalten, anderfalls geben sie WAHR zurück.

PHP stellt eine vollständige und mächtige Implementat von Ausdrücken bereit und, deren vollständige Dokumentation den Rahmen dieses Manuals sprengen würde. Die obigen Beispiele sollten Ihnen einen guten Eindruck darüber verschaffen, was Ausdrücke sind und wie man nützliche Ausdrücke konstruieren kann. Im Rest dieses Manuals werden wir *`ausdr`* schreiben, um ausdrücken, dass an dieser Stelle jeder gültige PHP-Ausdruck stehen kann.

Kapitel 10. Operatoren

Arithmetische Operatoren

Erinnerst du dich noch an die Grundrechenarten aus der Schule? Die arithmetischen Operatoren funktionieren genauso:

Tabelle 10-1. Arithmetic Operators

Beispiel	Name	Ergebnis
$a + $b	Addition	Summe von $a und $b.
$a - $b	Subtraktion	Differenz von $a und $b.
$a * $b	Multiplikation	Produkt von $a und $b.
$a / $b	Division	Quotient von $a und $b.
$a % $b	Modulus	Rest von $a geteilt durch $b.

Zuweisungsoperatoren

Der einfachste Zuweisungsoperator ist "=". Wahrscheinlich kommt man als erstes auf die Idee, ihn mit "ist gleich" zu bezeichnen. Das ist falsch. In Wirklichkeit bedeutet er, dass dem linken Operanden der Wert des rechten Operanden zugewiesen wird (man müsste ihn also mit "wird gesetzt auf den Wert von" übersetzen).

Der Wert eines Zuweisungs-Ausdruckes ist der zugewiesene Wert. D. h. der Wert des Ausdruckes "$a = 3" ist 3. Das erlaubt es, einige raffinierte Dinge zu tun:

```
$a = ($b = 4) + 5 // $a ist nun gleich 9 und $b wurde auf den Wert 4 gesetzt.
```

Zusätzlich zu dem oben vorgestellten Zuweisungsoperator "=" gibt es "kombinierte Operatoren" für alle binären, arithmetischen und String-Operatoren, die es erlauben, den Wert einer Variablen in einem Ausdruck zu benutzen, und dieser anschließend das Ergebnis des Ausdrucks als neuen Wert zuzuweisen. Beispiel:

```
$a = 3;
$a += 5; // setzt $a auf den Wert 8, als ob wir geschrieben haetten: $a = $a + 5;
$b = "Hallo ";
$b .= "Du!"; // setzt $b auf den Wert "Hallo Du!", aequivalent zu $b = $b . "There!";
```

Man beachte, dass die Zuweisung nur den Wert der Ursprungsvarialbe der neuen Variable zuweist (assignment by value)(sie "kopiert"), weshalb sich Änderungen an der einen Variablen nicht auf die andere auswirken werden. Das kann wichtig sein wenn man einen großen Array o. ä. in einer Schleife kopieren muss. PHP4 unterstützt 'assignement by reference' (Zuweisung als Verweis), mithilfe der `$var = &$othervar;`-Syntax, dies funktioniert jedoch nicht in PHP3. 'Assignement by reference' bedeutet, dass beide Variablen nach der Zuweisung die selben Daten repräsentieren. Nach der Zuweisung verändert eine Änderung der Variable `$var` auch den Inhalt der Variable `&$othervar;`, nichts wird kopiert, stattdessen wird ein "Verweis" auf die erste Variable angelegt.

Bit-Operatoren

Bit-Operatoren erlauben es, in einem Integer bestimmte Bits "ein- oder auszuschalten" (auf 0 oder 1 zu setzen).

Tabelle 10-2. Bit-Operatoren

Beispiel	Name	Ergebnis
$a & $b	Und	Die Bits sind in $a und in $b gesetzt.
$a | $b	Oder	Die Bits sind entweder in $a oder in $b gesetzt.
$a ^ $b	Entweder oder (Xor)	Die Bits sind in $a oder in $b gesetzt, aber nicht in beiden.
~ $a	Nicht	Die Bits, die in $a nicht gesetzt sind, sind gesetzt, und umgekehrt.
$a << $b	Nach links verschieben	Verschiebung der Bits von $a um $b Stellen nach links (Jede Stelle entspricht einer Mulitplikation mit zwei.
$a >> $b	Nach links verschieben	Verschiebt die Bits von $a um $b Stellen nach rechts (jede Stelle entspricht einer Division durch zwei).

Vergleichs-Operatoren

Vergleichs-Operatoren erlauben es - wie der Name schon sagt - zwei Werte zu vergleichen.

Tabelle 10-3. Vergleichsoperatoren

Beispiel	Name	Ergebnis
$a == $b	Gleich	Gibt "wahr" zurück, wenn von $a gleich $b ist.
$a === $b	Identisch	Gibt "wahr" zurück wenn $a gleich $b ist und beide vom gleichen Typ sind(funktioniert nur in PHP4)
$a != $b	Ungleich	Gibt "wahr" zurück, wenn $a nicht gleich $b ist.
$a < $b	Kleiner Als	Gibt "wahr" zurück, wenn $a kleiner als $b ist.
$a > $b	Größer Als	Gibt "wahr" zurück, wenn $a größer als $b ist.
$a <= $b	Kleiner Gleich	Gibt "wahr" zurück, wenn $a kleiner oder gleich $b ist.

Beispiel	Name	Ergebnis
$a >= $b	Größer Gleich	Gibt "wahr" zurück, wenn $a größer oder gleich $b ist.

Ein weiter Vergleichs-Operator ist der "?:"- oder Trinitäts-Operator. Er arbeitet genauso wie in C und vielen anderen Programmier-Sprachen.

```
(ausdr1) ? (ausdr2) : (ausdr3);
```

Dieser Ausdruch gibt *ausdr2* zurück, wenn *ausdr1* "wahr" zurückgibt, und *ausdr3*, wenn *ausdr1* "falsch" zurückgibt.

Fehler-Kontroll-Operatoren

PHP unterstützt einen Operator zur Fehlerkontrolle: Das @-Symbol. Stellt man das @ in PHP vor einen Ausdruck werden alle Fehlermeldungen, die von diesem Ausdruck erzeugt werden könnten, ignoriert.

Ist das track_errors-Feature aktiviert, werden alle Fehlermeldungen, die von diesem Ausdruck erzeugt werden, in der globalen Variablen $php_errormsg gespeichert. Da diese Variable mit jedem neuen Auftreten eines Fehlers überschrieben wird, sollte man sie möglichst bald nach Verwendung des Ausdrucks überprüfen, wenn man mit ihr arbeiten will.

```
<?php
/* Beabsichtigter SQL-Fehler (zusätzliches Anfuehrungszeichen): */
$res = @mysql_query( "select name, code from ’namelist" ) or
   die( "SQL-Anfrage fehlgeschlagen: Fehlermeldung: ’$php_errormsg’" );
?>
```

Siehe auch error_reporting().

Operatoren zur Programmausführung

PHP unterstützt einen Operator zur Ausführung externer Programme: Die sog. Backticks (“). Achtung: Die Backticks sind keine einfachen Anführungszeichen, sondern entsprechen auf den meisten Tastaturen den Tasten für die frz. Accents. PHP versucht, den Text zwischen den Backticks als Kommandozeilen-Befehl auszuführen. Zurückgegeben wird der die Ausgabe des aufgegebenen Programms (d. h. die Ausgabe wird nicht einfach gelöscht, sondern kann einer Variablen zugewiesen werden).

```
$output = ‘ls -al‘;
echo "<pre>$output</pre>";
```

Siehe auch system(), passthru(), exec(), popen(), und escapeshellcmd().

Inkrementierungs- bzw. Dekrementierungsoperatoren

PHP unterstützt Prä- und Post-Inkrementierungs- und Dekrementierungsoperatoren im Stil der Programmiersprache C.

Tabelle 10-4. Inkrement- und Dekrementoperatoren

Beispiel	Name	Auswirkung
++$a	Prä-Inkrement	Erhöht den Wert von $a um eins (inkrementiert $a), und gibt anschließend den neuen Wert von $a zurück.
$a++	Post-Inkrement	Gibt zuerst den aktuellen Wert von $a zurück und erhöht dann den Wert von $a um eins.
--$a	Prä-Dekrement	Vermindert den Wert von $a um eins (dekrementiert $a), und gibt anschließend den neuen Wert von $a zurück.
$a--	Post-Dekrement	Gibt zuerst den aktuellen Wert von $a zurück und erniedrigt dann den Wert von $a um eins.

Ein einfach Beispiel-Skript:

```
<?php
echo "<h3>Post-Inkrement</h3>";
$a = 5;
echo "Sollte 5 sein: " . $a++ . "<br>\n";
echo "Sollte 6 sein: " . $a . "<br>\n";

echo "<h3>Prae-Inkrement</h3>";
$a = 5;
echo "Sollte 6 sein: " . ++$a . "<br>\n";
echo "Sollte 6 sein: " . $a . "<br>\n";

echo "<h3>Post-Dekrement</h3>";
$a = 5;
echo "Sollte 5 sein: " . $a-- . "<br>\n";
echo "Sollte 4 sein: " . $a . "<br>\n";

echo "<h3>Praedekrement</h3>";
$a = 5;
echo "Sollte 4 sein: " . --$a . "<br>\n";
echo "Sollte 4 sein: " . $a . "<br>\n";
?>
```

Logische Operatoren

Tabelle 10-5. Logische Operatoren

Beispiel	Name	Ergebnis
$a and $b	Und	Wahr wenn sowohl $a als auch $b wahr ist.
$a or $b	Oder	Wahr wenn $a oder $b wahr ist.
$a xor $b	Entweder Oder	Wahr wenn entweder $a oder $b wahr ist, aber nicht beide.
! $a	Nicht	Wahr wenn $a nicht wahr ist.
$a && $b	Und	Wahr wenn sowohl $a als auch $b wahr ist.
$a || $b	Oder	Wahr wenn $a oder $b wahr ist.

Der Grund dafür, dass es je zwei unterschiedliche Operatoren für die "Und"- und die "Oder"-Verknüpfung gibt ist, dass die beiden Operatoren jeweils einen unterschiedliche Plätze in der Operator-Rangfolge haben. (siehe auch Operator-Rangfolge.)

Operator-Rangfolge

Die Operator-Rangfolge legt fest, wie "eng" ein Operator zwei Ausdrücke miteinander verbindet. Das Ergebnis des Ausdruckes `1 + 5 * 3` ist z. B. 16 und nicht 18, da der Mulitiplikations-Operator (*) in der Rangfolge höher steht als der Additions-Operator (+).

Die folgende Tabelle zeigt die Rangfolge der Operatoren, oben steht der Operator mit dem niedrigsten Rang.

Tabelle 10-6. Operator-Rangfolge

Associativity	Operator
links	,
links	or
links	xor
links	and
rechts	print
links	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
links	? :
links	||
links	&&
links	|
links	^
links	&
keine Richtung	== != ===
keine Richtung	< <= > >=
links	<< >>

Associativity	Operator
links	+ - .
links	* / %
rechts	! ~ ++ -- (int) (double) (string) (array) (object) @
rechts	[
keine Richtung	new

Zeichenketten-Operatoren

Es gibt in PHP zwei Operatoren für Zeichenketten. Der erste ist Vereinigungs-Operator. ('.'), dessen Rückgabewert eine Zeichenkette, zusammengesetzt aus dem rechten und dem linken Argument, ist. Der zweite ist der Vereinigungs-Zuweisungsoperator ('.='). Siehe Zuweisungs-Operatoren für weitere Informationen.

```
$a = "Hallo ";
$b = $a . "Welt!"; // $b enthaelt jetzt den Text "Hallo Welt!"

$a = "Hallo ";
$a .= "Welt!"; // $a enthält jetzt den Text "Hallo Welt!"
```

Kapitel 11. Kontroll-Strukturen

Jedes PHP-Skript besteht aus einer Reihe von Anweisungen. Eine Anweisung kann aus einem Funktions-Aufruf, einer Schleife, einer bedingten Anweisung oder einem Befehl, der nichts macht (eine leere Anweisung), bestehen. Jeder Befehl endet gewöhnlich mit einem Semikolon. Darüber hinaus können Befehle zu einer Anweisungsgruppe zusammengefasst werden, welche durch geschweifte Klammern begrenzt wird. Eine Anweisungsgruppe ist auch eine Anweisung. Die unterschiedlichen Arten von Anweisungen werden in diesem Abschnitt erläutert.

if

Der `if`-Befehl ist eine der wichtigsten Möglichkeiten vieler Programmier-Sprachen, PHP eingeschlossen. Er erlaubt die bedingte Ausführung von Programmteilen. PHP beinhaltet eine `if`-Struktur, die ähnlich der C-Programmiersprache ist:

```
if (ausdr)
    Anweisung
```

Wie im Abschnitt über Expressions / Ausdrücke beschrieben, wird ausdr auf seinen wirklichen Wertinhalt ausgewertet. Wenn *`ausdr`* `TRUE` entspricht, wird PHP Anweisung ausführen, falls nicht - sie also `FALSE` ist - wird Anweisung übergangen.

Das folgende Beispiel wird `a ist grösser als b` anzeigen, wenn *`$a`* grösser ist als *`$b`*:

```
if ($a > $b)
    print "a ist grösser als b";
```

Oft werden Sie die bedingte Ausführung von mehr als einer Anweisung wollen. Selbstverständlich ist es nicht erforderlich, jede Anweisung mit einer `if`-Bedingung zu versehen. Statt dessen können Sie mehrere Anweisungen in Gruppen zusammenfassen. Z.B. wird der folgende Programm-Code `a ist grösser als b` anzeigen, wenn *`$a`* grösser ist als *`$b`*. Danach wird der Wert von *`$a`* in *`$b`* gespeichert:

```
if ($a > $b) {
    print "a ist grösser als b";
    $b = $a;
}
```

If-Anweisungen können ohne Einschränkung innerhalb anderer `if`-Anweisungen definiert werden. Das ermöglicht ihnen völlige Flexibilität bei der bedingten Ausführung verschiedenster Programmteile.

else

Häufig ist es erforderlich, eine Anweisung auszuführen, wenn eine bestimmte Bedingung erfüllt ist und eine andere Anweisung, falls sie nicht erfüllt ist. Dafür gibt es `else`. `Else` erweitert eine `if`-Anweisung um die Ausführung von Anweisungen, sobald der Ausdruck der `if`-Anweisung als `FALSE` angesehen wird. Der folgende Code wird z.B. `a ist grösser als b` anzeigen, wenn *$a* grösser ist als *$b*, anderenfalls `a ist NICHT grösser als b`:

```
if ($a > $b) {
    print "a ist grösser als b";
} else {
    print "a ist NICHT grösser als b";
}
```

Die `else`-Anweisung wird nur ausgeführt, wenn der `if`-Ausdruck als `FALSE` ausgewertet wurde und wenn bei vorhandenen `elseif`-Ausdrücken diese ebenfalls `FALSE` sind (siehe elseif).

elseif

`Elseif` ist, wie der Name schon sagt, eine Verbindung von `if` und `else`. Wie `else` erweitert sie eine `if`-Anweisung um die Ausführung anderer Anweisungen, sobald die normale `if`-Bedingung als `FALSE` angesehen wird. Anders als bei `else` wird die Ausführung dieser anderen Anweisungen nur durchgeführt, wenn die bei `elseif` angegebene alternative Bedingung als `TRUE` angesehen wird. Der folgende Code wird z.B. `a ist grösser als b`, `a ist gleich b` oder `a ist kleiner als b` ausgeben:

```
if ($a > $b) {
    print "a ist grösser als b";
} elseif ($a == $b) {
    print "a ist gleich b";
} else {
    print "a ist kleiner als b";
}
```

Es kann mehrere `elseif`-Anweisungen innerhalb einer `if`-Anweisung geben. Die erste `elseif`-Bedingung (falls vorhanden), die `TRUE` ist, wird ausgeführt. In PHP kann man auch 'else if' schreiben (zwei Wörter). Das Verhalten ist identisch zu 'elseif' (ein Wort), genau wie in C.

Die `elseif`-Anweisung wird nur ausgeführt, wenn die vorausgehende `if`-Bedingung sowie jede vorherige `elseif`-Bedingung `FALSE` ist und die aktuelle `elseif`-Bedingung `TRUE` ist.

Alternative Syntax für Kontroll-Strukturen

PHP bietet eine alternative Syntax für einige seiner Kontroll-Strukturen, als da sind `if`, `while`, `for`, `foreach` und `switch`. Immer muss die öffnende Klammer durch einen Doppelpunkt ":" und die

schließende Klammer durch ein `endif;`, `endwhile;`, `endfor;`, `endforeach;` bzw. `endswitch;` ersetzt werden.

```
<?php if ($a == 5): ?>
    A ist gleich 5
<?php endif; ?>
```

Im obigen Beispiel ist der HTML-Bereich eingebettet in eine `if`-Anweisung mit alternativer Syntax. Der HTML-Bereich wird nur ausgegeben, wenn $a gleich 5 ist.

Die alternative Syntax kann auch auf `else` und `elseif` angewendet werden. Es folgt eine `if`-Struktur mit `elseif` und `else` im alternativen Format:

```
if ($a == 5):
    print "a ist gleich 5";
    print "...";
elseif ($a == 6):
    print "a ist gleich 6";
    print "!!!";
else:
    print "a ist weder 5 noch 6";
endif;
```

Siehe auch while, for und if für weitere Beispiele.

while

`While`-Schleifen sind die einfachste Form von Schleifen in PHP. Sie funktionieren genau wie in C. Die Grundform einer `while`-Anweisung lautet:

```
while (ausdr) Anweisung
```

Die Bedeutung einer `while`-Anweisung ist einfach. Sie weist PHP an, einen in ihr eingebetteten Befehl so lange zu wiederholen, bis die `while`-Bedingung `TRUE` geworden ist. Der Wert der Bedingung wird immer am Anfang der Schleife geprüft. Wird der Wert während der Ausführung der Befehle innerhalb der `while`-Schleife geändert, endet die Ausführung dieses Befehls-Blocks nicht vor einem neuem Schleifen-Durchlauf (Iteration). Jeder Schleifendurchlauf ist eine Iteration. Falls die `while`-Bedingung bereits zu Beginn `FALSE` ist, werden die Anweisungen der `while`-Schleife nicht ein einziges Mal durchlaufen.

Wie bei der `if`-Anweisung kann man mehrere Befehle innerhalb einer Schleife angeben, indem man sie mit geschweiften Klammern umschliesst oder die alternative Syntax gebraucht:

```
while (ausdr): Anweisung ... endwhile;
```

Die folgenden Beispiele sind identisch; beide geben Zahlen von 1 bis 10 aus:

```
/* Beispiel 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  /* es wird erst $i ausgegeben,
                    bevor der Wert erhöht wird
                    (Post-Inkrement) */
}

/* Beispiel 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

`Do..while`-Schleifen sind den `while`-Schleifen sehr ähnlich, außer dass die Erfüllung des Ausdrucks am Ende jedes Durchlaufs geprüft wird (statt am Anfang). Der Hauptunterschied zu gewöhnlichen `while`-Schleifen ist, dass der erste Schleifen-Durchlauf bei `do..while` in jedem Fall statt findet, wogegen es bei `while`-Schleifen durchaus passieren kann, dass die Schleife nie durchlaufen wird, wenn die am Anfang zu prüfende Bedingung schon zu Beginn `FALSE` ist.

Es gibt nur eine Syntax für `do..while`-Schleifen:

```
$i = 0;
do {
    print $i;
} while ($i>0);
```

Die obige Schleife wird genau einmal durchlaufen, da nach der ersten Wiederholung die Erfüllung der Bedingung geprüft wird. Da diese aber nicht erfüllt, also `FALSE` ist ($i ist nicht grösser als 0), wird die Schleifenausführung beendet.

Erfahrene C-Anwender kennen auch die Möglichkeit, Programm-Blöcke mit `do..while`(0) einzuschliessen und dann die `break` Anweisung zu benutzen. Der folgende Programm-Ausschnitt zeigt dies:

```
do {
    if ($i < 5) {
        print "i ist nicht gross genug";
        break;
    }
```

```
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i ist ok";

    ...bearbeite i...

} while(0);
```

Es ist nicht weiter tragisch, wenn Sie dieses Beispiel nicht oder nur zum Teil verstehen. Sie können auch ohne dieses Feature effektive PHP-Programme und Skripte schreiben.

for

`For`-Schleifen sind die komplexesten Schleifen in PHP. Sie funktionieren wie ihr Gegenstück in C. Die Syntax einer `for`-Schleife sieht so aus:

```
for (ausdr1; ausdr2; ausdr3) Anweisung
```

Der erste Parameter (*`ausdr1`*) wird beim Schleifenbeginn geprüft bzw. ausgeführt (ohne jegliche Vorbedingung).

Zu Beginn jedes Durchlaufs wird nun *`ausdr2`* geprüft. Wenn dieser `TRUE` ist, fährt die Schleife weiter fort mit der Ausführung der nachfolgenden Befehle. Wenn das Ergebnis `FALSE` lautet, wird die Schleife beendet.

Am Ende jedes Durchlaufs wird nun auch noch *`ausdr3`* geprüft / ausgeführt.

Jeder der Parameter kann leer sein (optional). Ist *`ausdr2`* leer, wird die Schleife unbestimmt oft durchlaufen, da PHP ihn als `TRUE` wertet (wie in C). Das ist nicht so sinnlos, wie Sie vielleicht glauben, weil man häufig eine Schleife erst durch eine bedingte `break` -Anweisung statt durch eine unwahr werdende `for`-Bedingung beenden möchte.

Beachten Sie die folgenden Beispiele. Alle geben Zahlen von 1 bis 10 aus:

```
/* Beispiel 1 */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* Beispiel 2 */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}
```

```
/* Beispiel 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* Beispiel 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Selbstverständlich sieht das erste (oder das vierte) Beispiel am besten aus, aber Sie werden noch feststellen, dass es oftmals ganz nützlich sein kann, leere Parameter in `for`-Schleifen zu verwenden.

PHP unterstützt auch bei `for`-Schleifen die alternative "Doppelpunkt-Syntax".

```
for (ausdr1; ausdr2; ausdr3): Anweisung; ...; endfor;
```

Andere Sprachen haben für das Durchlaufen eines Hash´s oder Arrays eine `foreach`-Anweisung. PHP 3 hat dies nicht; im Gegensatz zu PHP 4 (vgl. foreach). In PHP 3 kann man dafür eine Kombination von while mit der list()- und each()-Funktion einsetzen. Beispiele finden Sie in der Dokumentation zu diesen Funktionen.

foreach

PHP 4 (nicht PHP 3) enthält eine `foreach`-Funktion, genau wie Perl und andere Sprachen. Diese ermöglicht es, auf einfache Weise ein Array zu durchlaufen. Es gibt zwei Syntax-Formen; die zweite ist eine unbedeutende aber sinnvolle Erweiterung der ersten Syntax:

```
foreach(array_expression as $value) Anweisung
foreach(array_expression as $key => $value) Anweisung
```

Die erste Form durchläuft das `array_expression`-Array. Bei jedem Durchgang wird der Wert des aktuellen Elements `$value` zugewiesen und der interne Array-Zeiger um 1 erhöht. Dadurch wird beim nächsten Durchgang automatisch das nächste Element ausgewertet.

Die zweite Form arbeitet genauso, ausser dass bei jedem Durchlauf auch der aktuelle Schlüssel der Variablen `$key` zugewiesen wird.

Anmerkung: Sobald `foreach` zum ersten Mal ausgeführt wird, wird der interne Array-Pointer automatisch auf das erste Element des Arrays zurück gesetzt. Das heisst, dass Sie nicht reset() vor einem Durchlauf von `foreach` aufrufen müssen.

Anmerkung: Beachten Sie auch, dass `foreach` mit einer Kopie des angegebenen Arrays arbeitet. Deshalb wird auch der Array-Pointer nicht wie bei dem each-Konstrukt verändert.

Beachten Sie, dass die folgenden Beispiele in ihrer Funktionalität identisch sind:

```
      reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Wert: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Wert: $value<br>\n";
}
```

Auch hier funktioniert alles gleich:

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Schlüssel: $key; Wert: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Schlüssel: $key; Wert: $value<br>\n";
}
```

Noch einige Beispiele, die die Anwendung verdeutlichen:

```
/* foreach Beispiel 1: Nur der Wert */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
    print "Aktueller Wert von \$a: $v.\n";
}

/* foreach Beispiel 2: Wert (mit Ausgabe des Array-Schlüssels) */

$a = array (1, 2, 3, 17);

$i = 0; /* nur für Anschauungs-Zweck */
```

```
foreach($a as $v) {
    print "\$a[$i] => $k.\n";
    $i++;
}

/* foreach Beispiel 3: Schlüssel und Wert */

$a = array (
    "eins" => 1,
    "zwei" => 2,
    "drei" => 3,
    "siebzehn" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}
```

break

`Break` bricht die Ausführung der aktuellen `for`, `while` oder `switch` Anweisungs-Sequenz ab.

Einem `break` kann optional ein nummerisches Argument angehängt werden, das die Anzahl der abzubrechenden Befehls-Sequenzen enthält.

```
$arr = array ('eins', 'zwei', 'drei', 'stop', 'vier');
while (list ( , $val) = each ($arr)) {
    if ($val == 'stop') {
        break;  /* Man kann hier auch 'break 1;' schreiben. */
    }
    echo "$val<br>\n";

/* Benutzung des optionalen Argumentes. */
$i = 0;
    while (++$i) {
        switch ($i) {
            case 5:
                echo "Bei 5<br>\n";
                break 1;  /* Beendet nur switch. */
            case 10:
                echo "Bei 10; quitting<br>\n";
                break 2;  /* Beendet switch und while. */
            default:
                break;
    }
}
```

continue

`Continue` wird innerhalb von Schleifen gebraucht. Die Schleife wird an der aktuellen Stelle abgebrochen und es wird der nächste Durchlauf begonnen.

`Continue` kann optional ein nummerisches Argument erhalten, das die Anzahl der zu überspringenden Schleifendurchläufe beinhaltet.

```
while (list ($key, $value) = each ($arr)) {
    if ($key % 2) { // überspringe ungerade Werte
        continue;
    }
    tue_was_mit_ungerade ($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Aussen<br>\n";
    while (1) {
        echo "  Mitte<br>\n";
        while (1) {
            echo "  Innen<br>\n";
            continue 3;
        }
        echo "Das wird nie ausgegeben.<br>\n";
    }
    echo "Dies auch nicht.<br>\n";
}
```

switch

Die `switch`-Anweisung ist gleichbedeutend einer Reihe von `if`-Anweisungen mit dem gleichen Parameter. Häufig wollen Sie ein und die selbe Variable (bzw. den selben Ausdruck) mit verschiedensten Werten vergleichen und in Abhängigkeit vom Auswertungsergebnis verschiedene Programmteile ausführen. Genau das ermöglicht die `switch`-Anweisung.

Die folgenden 2 Beispiele zeigen 2 verschiedene Wege, das gleiche zu bewirken; eins gebraucht mehrere `if`-Befehle, das andere eine `switch`-Anweisung:

```
if ($i == 0) {
    print "i ist gleich 0";
}
if ($i == 1) {
    print "i ist gleich 1";
}
if ($i == 2) {
    print "i ist gleich 2";
}

switch ($i) {
    case 0:
        print "i ist gleich 0";
```

```
        break;
    case 1:
        print "i ist gleich 1";
        break;
    case 2:
        print "i ist gleich 2";
        break;
}
```

Es ist wichtig, die Ausführung einer `switch`-Anweisung zu verstehen, um Fehler zu vermeiden. Die `switch`-Anweisung wird Zeile für Zeile (also Anweisung für Anweisung) abgearbeitet. Zu Beginn wird nichts ausgeführt. Erst wenn bei einem `case`-Teil eine Entsprechung zum `switch`-Ausdruck vorliegt, werden die darin enthaltenen Befehle ausgeführt. PHP fährt dann mit der Abarbeitung des restlichen Codes innerhalb des `switch`-Blocks fort (oder bis zum ersten Auftreten einer `break`-Anweisung). Ohne `break` am Ende eines case-Teils werden also noch die folgenden case-Blöcke ausgeführt. Z.B.:

```
switch ($i) {
    case 0:
        print "i ist gleich 0";
    case 1:
        print "i ist gleich 1";
    case 2:
        print "i ist gleich 2";
}
```

Wenn hier $i gleich 0 ist, würde PHP alle print-Anweisungen ausführen. Ist $i gleich 1, würden die letzten beiden print-Befehle ausgeführt und wenn $i = 2 ist, würde nur der letzte print-Befehl ausgeführt. Deshalb ist es wichtig, bei der ersten gefundenen Übereinstimmung eine `break`-Anweisung zu setzen (abgesehen von bestimmten Fällen, wo genau dieses Verhalten gefordert wird).

Bei einer `switch`-Anweisung wird die Bedingung also nur einmal überprüft und das Ergebnis mit jeder `case`-Anweisung verglichen. Bei einem `elseif`-Befehl wird die Bedingung neu geprüft. Eine `switch`-Anweisung kann schneller als mehrere `if`-Befehle sein, z.B. bei komplizierteren Bedingungen als einem einfachen Vergleich.

Der Anweisungsteil von case kann auch leer sein. Dann wird die Kontrolle einfach an den nächsten case-Teil übergeben.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i ist kleiner als 3 aber nicht negativ";
        break;
    case 3:
        print "i ist gleich 3";
}
```

Ein Spezialfall ist die Anweisung default. Diese trifft auf alles zu, was nicht von den voranstehenden case-Ausdrücken erfasst wurde, wie z.B.:

```
switch ($i) {
    case 0:
        print "i ist gleich 0";
        break;
    case 1:
        print "i ist gleich 1";
        break;
    case 2:
        print "i ist gleich 2";
        break;
    default:
        print "i ist weder 0, 1 noch 2";
}
```

Der `case`-Ausdruck kann eine Prüfung einfacher Typen sein, also von Integer- oder Fließkomma-Zahlen oder von Strings / Zeichenketten. Arrays oder Objekte können nicht benutzt werden, es sei denn, sie werden auf einfache Typen herunter gebrochen.

Die alternative Syntax gilt auch für switch-Sequenzen. Mehr Informationen dazu unter Alternative Syntax für Kontroll-Strukturen .

```
switch ($i):
    case 0:
        print "i ist gleich 0";
        break;
    case 1:
        print "i ist gleich 1";
        break;
    case 2:
        print "i ist gleich 2";
        break;
    default:
        print "i ist weder 0, 1 noch 2";
endswitch;
```

require()

Der require()-Befehl setzt an seine Stelle den Inhalt der angegebenen Datei (ähnlich dem #include von C).

Wenn die PHP-Konfigurations-Option "URL fopen wrappers" auf enabled (per default) steht, können Sie für die mittels require() eingebundene Datei statt einer lokalen Pfadangabe auch eine URL angeben. Sehen Sie hierzu unter Remote-Dateien und fopen() für weitere Informationen nach.

Wichtig: PHP kehrt zu Beginn der per include() oder require() eingebundenen Dateien vom PHP- in den HTML-Modus und am Schluss der Datei wieder vom HTML- in den PHP-Modus zurück. Falls innerhalb dieser Dateien also PHP-Code ausgeführt werden soll, muss dieser eingeschlossen werden von gültigen PHP-Start- und PHP-Ende-Marken (siehe gültige PHP-Start- und Ende-Tags).

require() ist keine PHP-Anweisung, sondern eine spezielle Sprachanweisung. Sie ist anderen Regeln unterworfen als Funktionen. Einerseits unterliegt sie keinen Kontroll-Srukturen, andererseits gibt sie keinen Wert zurück. Der Versuch, von einem require()-Aufruf einen Rückgabewert zu erhalten, führt zu einem Parse-Fehler.

Anders als include() wird require() *immer* die angegebene Datei einlesen, *auch dann, wenn die Programmzeile, in der sie steht, nicht ausgeführt wird.* Wenn Sie eine Datei nur bedingt einlesen wollen, benutzen Sie include(). Die bedingte Anweisung würde require() nicht davon abhalten, die Datei zu laden.

Innerhalb von Schleifen tritt der Effekt auf, dass, obwohl die den require()-Befehl enthaltende Zeile mehrfach angesprungen wird, die entsprechende Datei trotzdem nur genau einmal eingelesen (ausgeführt) wird.

Sie können also require() nicht innerhalb von Programm-Schleifen einsetzen. Deshalb, und wenn Sie verschiedene Dateien einlesen wollen, müssen Sie in Schleifen den Befehl include() benutzen.

```
require ('header.inc');
```

Wird eine Datei per require() eingebunden, dann "erbt" der darin enthaltene Code die bis zur Skriptzeile mit der require()-Anweisung definierten Variablen und deren Inhalt. Alle innerhalb der eingebundenen Datei Definitionen von Variablen stehen ab dieser Zeile zur Verfügung. Steht die require()-Anweisung innerhalb einer Funktion, wird der darin stehende Code behandelt, als ob er innerhalb der Funktion stehen würde.

Wird die mittels require() einzubindende Datei über HTTP (fopen wrapper) aufgerufen und sieht der Ziel-Server die Ziel-Datei als PHP-Code ansieht, dann werden die Variablen an die einzubindende Datei wie bei einem HTTP GET übergeben. Das entspricht nicht genau der Einbindung einer lokalen Datei, da das Skript auf dem entfernten Server ausgeführt wird und nur die Ergebnisse in das lokale Skript eingebunden werden.

```
/* Dieses Beispiel geht davon aus, dass ein beliebiger Server zum
 * Parsen von *.php-Dateien konfiguriert ist. Also meint 'funkt' hier,
 * dass die Variablen $vareins und $varzwei innerhalb der angeforderten
 * Datei vorhanden sind. */

/* Funkt. nicht; file.txt wird vom besagten Server nicht geparst. */
require ("http://someserver/file.txt?vareins=1&varzwei=2");

/* Funkt. nicht; schaut nach einer lokalen Datei namens
/* 'file.php?varone=1&vartwo=2' */
require ("file.php?vareins=1&varzwei=2");

/* Funkt. */
require ("http://someserver/file.php?vareins=1&varzwei=2");

$vareins = 1;
$varzwei = 2;
require ("file.txt");    /* Funkt. */
```

```
require ("file.php");    /* Funkt. */
```

In PHP 3 ist es möglich, eine `return`-Anweisung innerhalb einer per require() eingebundenen Datei auszuführen, solange diese Anweisung im Hauptteil dieser Datei steht. Die `return`-Anweisung darf nicht in Blöcken (also innerhalb von geschwungenen Klammern "{}" stehen. In PHP 4 existiert diese Möglichkeit nicht mehr. Wenn Sie diese dennoch benötigen, sehen Sie unter include() nach.

Siehe auch include(), require_once(), include_once(), readfile() und virtual().

include()

Die include()-Anweisung liest die angegebene Datei ein und wertet sie aus.

Wenn die PHP-Konfigurations-Option "URL fopen wrappers" auf enabled (per default) steht, können Sie für die mittels include() eingebundene Datei statt einer lokalen Pfadangabe auch eine URL angeben. Sehen Sie hierzu unter Remote-Dateien und fopen() für weitere Informationen nach.

Wichtig ist, dass beim include()- oder require()-Befehl vom PHP-Parsing-Modus in den HTML-Modus geschaltet wird und bei Rückkehr in das aufrufende Skript wieder zurück vom HTML- in den PHP-Modus. Deshalb muss jeder PHP-Code innerhalb der eingebundenen Dateien umschlossen werden von gültigen PHP-Start- und Ende-Tags.

Das passiert jedes Mal, sobald die include()-Anweisung auftritt. Deshalb können Sie include() auch sehr gut innerhalb von Schleifen verwenden, um eine Anzahl unterschiedlicher Dateien einzubinden.

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
    include $files[$i];
}
```

include() unterscheidet sich von require() dadurch, dass die include-Anweisung jedesmal neu ausgewertet wird, sobald sie auftritt (und nur zur Ausführungszeit). Dagegen wird die require()-Anweisung beim ersten Auftreten mit der angegebenen Datei ersetzt, egal ob sie einzubinden ist oder nicht (innerhalb von bedingten Anweisungen, z.B. bei if auch dann, wenn das ausgewertete Argument `FALSE` ergeben hat).

Da include() ein spezieller Sprach-Konstrukt ist, müssen Sie ihn innerhalb einer bedingten Anweisung in einen Anweisungsblock setzen.

```
/* Das ist falsch und führt nicht zum gewünschten Ergebnis. */

if ($Bedingung)
    include($diesedatei);
else
    include($anderedatei);

/* Diese ist korrekt. */

if ($Bedingung) {
```

```
    include($diesedatei);
} else {
    include($anderedatei);
}
```

Sowohl in PHP 3 als auch in PHP 4 ist es möglich, eine `return`-Anweisung innerhalb einer include()eten Datei anzugeben, um die Ausführung dieser Datei abzubrechen und zum aufrufenden-Skript zurück zu kehren. Es gibt aber ein paar Unterschiede in der Arbeitsweise. Der erste ist, dass bei PHP 3 die `return`-Anweisung nicht innerhalb eines Blocks auftreten darf, es sei denn, es ist ein Funktions-Block. In diesem Fall gilt `return` für diese Funktion und nicht für die ganze Datei. In PHP 4 gibt es diese Beschränkung nicht. PHP 4 erlaubt ihnen auch die Rückgabe von Werten bei include()eten Dateien. Sie können den Wert des include()-Aufrufs nutzen, als wenn Sie eine Funktion aufgerufen hätten. Das wird in PHP 3 einen Parse-Error ergeben.

Beispiel 11-1. include() in PHP 3 und PHP 4

Beachten Sie, dass die folgende Datei, genannt `test.inc` im gleichen Verzeichnis wie die Hauptdatei stehen muss:

```
<?php
echo "Vor dem Return <br>\n";
if (1) {
    return 27;
}
echo "Nach dem Return <br>\n";
?>
```

Die Datei `main.html` enthält folgendes:

```
<?php
$retval = include ('test.inc');
echo "Datei gibt zurück: '$retval'<br>\n";
?>
```

Sobald `main.html` in PHP 3 aufgerufen wird, wird ein Parse-Error in Zeile 2 angezeigt; Sie können in PHP 3 also keinen Rückgabewert von include() erhalten. In PHP 4 wird das Ergebnis sein:

```
Vor dem Return
Datei gibt zurück: '27'
```

Die Datei `main.html` soll nun folgendes enthalten:

```
<?php
include ('test.inc');
echo "Zurück in main.html<br>\n";
?>
```

In PHP 4 wird die Ausgabe sein:

```
Vor dem Return
Zurück in main.html
```

PHP 3 wird dagegen folgendes ausgeben:

```
Vor dem Return
```

```
27Zurück in main.html

Parse error: parse error in /home/torben/public_html/phptest/main.html on li-
ne 5
```

Der obige Parse-Error ist das Ergebnis der Tatsache, dass die `return`-Anweisung innerhalb eines Nicht-Funktions-Blocks von `test.inc` steht. Wenn das return ausserhalb diese Blocks zu stehen kommt, wird die Ausgabe wie folgt aussehen:

```
Before the return
27Back in main.html
```

Die '27' entsteht aus der Tatsache, dass PHP 3 keine solchen Rückgabewerte unterstützt.

Wird eine Datei per include() eingebunden, dann "erbt" der darin enthaltene Code die bis zur Skriptzeile mit der include()-Anweisung definierten Variablen und deren Inhalt. Alle innerhalb der eingebundenen Datei Definitionen von Variablen stehen ab dieser Zeile zur Verfügung. Steht die include()-Anweisung innerhalb einer Funktion, wird der darin stehende Code behandelt, als ob er innerhalb der Funktion stehen würde.

Wird die mittels include() einzubindende Datei über HTTP (fopen wrapper) aufgerufen und sieht der Ziel-Server die Ziel-Datei als PHP-Code ansieht, dann werden die Variablen an die einzubindende Datei wie bei einem HTTP GET übergeben. Das entspricht nicht genau der Einbindung einer lokalen Datei, da das Skript auf dem entfernten Server ausgeführt wird und nur die Ergebnisse in das lokale Skript eingebunden werden.

```
/* Dieses Beispiel geht davon aus, dass ein beliebiger Server zum
 * Parsen von *.php-Dateien konfiguriert ist. Also meint 'funkt' hier,
 * dass die Variablen $vareins und $varzwei innerhalb der angeforderten
 * Datei vorhanden sind. */

/* Funkt. nicht; file.txt wird vom besagten Server nicht geparst. */
include ("http://someserver/file.txt?vareins=1&varzwei=2");

/* Funkt. nicht; schaut nach einer lokalen Datei namens
/* 'file.php?varone=1&vartwo=2' */
include ("file.php?vareins=1&varzwei=2");

/* Funkt. */
include ("http://someserver/file.php?vareins=1&varzwei=2");

$vareins = 1;
$varzwei = 2;
include ("file.txt");    /* Funkt. */
include ("file.php");    /* Funkt. */
```

Lassen Sie sich nicht durch die Tatsache irritieren, dass Sie Dateien auch über HTTP per require oder include einbinden können (vgl. Remote files). Das oben Gesagte behält seine Gültigkeit.

Siehe auch require(), require_once(), include_once(), readfile() und virtual().

require_once()

Die require_once()-Anweisung ersetzt sich selbst durch die angegebene Datei (ähnlich der C-Preprozessor-Anweisung `#include`), funktioniert also ähnlich wie die require()-Anweisung. Der Hauptunterschied dazu liegt in der Tatsache, dass bei require_once() der einzubindende Code genau einmal in das Skript eingefügt wird.

Sie erzeugen beispielsweise die folgenden zwei Include-Dateien `utils.inc` und `foolib.inc`:

Beispiel 11-2. utils.inc

```
<?php
define(PHPVERSION, floor(phpversion()));
echo "GLOBALE SIND GUT\n";
function guterTee() {
    return "Oolong-Tee schmeckt gut!";
}
?>
```

Beispiel 11-3. foolib.inc

```
<?php
require ("utils.inc");
function zeigeVar($var) {
    if (PHPVERSION == 4) {
        print_r($var);
    } else {
        var_dump($var);
    }
}

// es folgen weiter Funktionen ...
?>
```

Nun schreiben Sie ein Skript `cause_error_require.php`:

Beispiel 11-4. cause_error_require.php

```
<?php
require("foolib.inc");
/* das Folgende erzeugt einen Fehler */
require("utils.inc");
$foo = array("1",array("complex","quaternion"));
echo "dies erfordert utils.inc, das auch\n";
echo "in foolib.inc erforderlich ist\n";
echo "Aufruf von guterTee: ".guterTee()."\n";
echo "Ausgabe foo: \n";
showVar($foo);
?>
```

Wenn Sie letzteres starten, wird folgende Ausgabe erzeugt (gilt für PHP 4.01pl2):

```
GLOBALE SIND GUT
GLOBALE SIND GUT

Fatal error:  Cannot redeclare causeerror() in utils.inc on line 5
```

Durch Umschreiben von `foolib.inc` und `cause_errror_require.php` (Gebrauch von require_once() statt require()) und Umbenennung des letzten Skriptes zu `avoid_error_require_once.php` haben wir nun:

Beispiel 11-5. foolib.inc (fixed)

```
...
require_once("utils.inc");
function showVar($var) {
...
```

Beispiel 11-6. avoid_error_require_once.php

```
...
require_once("foolib.inc");
require_once("utils.inc");
$foo = array("1",array("complex","quaternion"));
...
```

Beim nachfolgenden Aufruf wird folgende Ausgabe erzeugt (bei PHP 4.0.1pl2):

```
GLOBALE SIND GUT
dies erfordert utils.inc, das auch
in foolib.inc erforderlich ist
Aufruf von guterTee: Oolong-Tee schmeckt gut!.
Ausgabe foo:
Array
(
    [0] => 1
    [1] => Array
        (
            [0] => complex
            [1] => quaternion
        )

)
```

Beachten Sie, dass analog zum Verhalten von `#include` im C-Preprozessor diese Anweisung zur Zeit der Skript-"Compilierung" ausgewertet wird; sollten Sie also z.B. das Skript vor dessen Ausführung parsen und bei der Ausführung keinen Gebrauch machen wollen von dynamisch einzuschließenden Teilen, sollten Sie include_once() oder include() verwenden.

Weiter Beispiele zu require_once() und include_once() können Sie dem PEAR-Code entnehmen, welcher im aktuellsten PHP-Source-Code enthalten ist.

Siehe auch: require(), include(), include_once(), get_required_files(), get_included_files(), readfile(), und virtual().

include_once()

Die Anweisung include_once() schließt zur Ausführungszeit das angegebene Skript ein und führt es aus. Dies ist ähnlich der include()-Anweisung; mit dem Unterschied, dass einmal "includierter" Code nicht nochmals eingebunden wird.

Wie in der Beschreibung zu require_once() bereits angemerkt, soll include_once() in den Fällen verwendet werden, wo die selbe Datei mehr als einmal während der Ausführung eingebunden werden soll und Sie Wiederholungen bei den Funktions-Definitionen und Wertzuweisungen zu Variablen usw. vermeiden müssen.

Weiter Beispiele zu require_once() und include_once() können Sie dem PEAR-Code entnehmen, welcher im aktuellsten PHP-Source-Code enthalten ist.

include_once() steht ab PHP 4.01pl2 zur Verfügung.

Siehe auch: require(), include(), require_once(), get_required_files(), get_included_files(), readfile() und virtual().

Kapitel 12. Funktionen

Vom Nutzer definierte Funktionen

Eine Funktion kann wie folgt definiert werden:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Beispielfunktion.\n";
    return $retval;
}
```

Jeder beliebige korrekte PHP-Code kann in einer Funktion vorkommen, sogar andere Funktionen und Klassen- Definitionen.

In PHP3 müssen Funktionen definiert sein, bevor man auf sie verweist. In PHP4 ist das nicht mehr erforderlich.

Funktionsparameter

Mit einer Parameterliste kann man Informationen an eine Funktion übergeben. Die Parameterliste ist eine kommagetrennte Liste von Variablen und/oder Konstanten.

PHP unterstützt die Weitergabe von Parametern als Werte (das ist der Standard), als Verweise , und als Vorgabewerte . Die Übergabe einer beliebigen Parameterzahl ist nicht möglich, einen ähnlichen Effekt kann man aber durch die Übergabe von Arrays erreichen.

```
function rechne_array($eingabe) {
    echo "$eingabe[0] + $eingabe[1] = ", $eingabe[0]+$eingabe[1];
}
```

Verweise als Parameter übergeben

Normalerweise werden den Funktionen Werte als Parameter übermittelt. Wenn man den Wert dieses Parameters innerhalb der Funktion ändert, bleibt der Parameter ausserhalb der Funktion unverändert. Wollen Sie aber genau das erreichen, dann müssen Sie die Parameter als Verweise übergeben.

Wenn eine Funktion einen Parameter generell als Verweis behandeln soll, setzt man ein kaufmännisches Und (&) in der Funktionsdefinition vor den Parameternamen:

```
function fuege_etwas_anderes_an (&$string) {
    $string .= 'und nun zu etwas völlig anderem.';
}
$str = 'Dies ist ein String, ';
fuege_etwas_anderes_an ($str);
echo $str; // Ausgabe 'Dies ist ein String, und nun zu etwas völlig anderem.'
```

Wenn Sie eine Variable andererseits als Verweis an eine Funktion übergeben wollen, die dies nicht tut, können Sie das kaufmännische Und (&) auch beim Aufruf der Funktion verwenden:

```
function bla ($fasel) {
    $fasel .= ' und nun zu etwas völlig anderem.';
}
$str = 'Dies ist ein String, , ';
bla ($str);
echo $str;  // outputs 'Dies ist ein String, '
bla (&$str);
echo $str;  // outputs 'Dies ist ein String, und nun zu etwas völlig anderem.'
```

Vorgabewerte für Parameter

Eine Funktion kann C++-artige Vorgabewerte für skalare Parameter wie folgt definieren:

```
function machkaffee ($typ = "Cappucino") {
    return "Ich mache eine Tasse $typ.\n";
}
echo machkaffee ();
echo machkaffee ("Espresso");
```

Die Ausgabe von diesem kleinen Skript ist:

```
Ich mache eine Tasse Cappucino.
Ich mache eine Tasse Espresso.
```

Der Vorgabewert muß eine Konstante sein, darf also (zum Beispiel) keine Variable oder Element einer Klasse sein.

Bitte beachten Sie, dass alle Vorgabewerte rechts von den Nicht-Vorgabeparametern stehen sollten; - sonst wird es nicht funktionieren. Betrachten Sie folgendes Beispiel:

```
function mach_joghurt ($typ = "rechtsdrehendes", $geschmack) {
    return "Mache einen Becher $typ $geschmack-joghurt.\n";
}

echo mach_joghurt ("Brombeer");   // arbeitet nicht wie erwartet
```

Die Ausgabe dieses Beispiels ist::

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Mache einen Becher Brombeer-joghurt.
```

Nun vergleichen Sie bitte oberes Beispiel mit folgendem:

```
function mach_joghurt ($geschmack, $typ = "rechtsdrehendes") {
    return "Mache einen Becher $typ $geschmack-joghurt.\n";
}

echo mach_jogurt ("Brombeer");   // arbeitet wie erwartet.
```

... und jetzt ist die Ausgabe:

```
Mache einen Becher rechtsdrehendes Brombeer-Joghurt.
```

Rückgabewerte

Sie können Werte mit dem optionalen Befehl "return" zurückgeben. Es können Variablen jedes Typs zurückgegeben werden, auch Listen oder Objekte.

```
function quadrat ($zahl) {
    return $zahl * $zahl;
}
echo quadrat (4);   // gibt '16' aus.
```

Es ist leider nicht möglich, mehrere Werte von einer Funktion zurückgeben zu lassen. Ein ähnliches Resultat kann man aber durch die Rückgabe von Listen bekommen.

```
function kleine_zahlen() {
   return array (0, 1, 2);
}
list ($null, $eins, $zwei) = kleine_zahlen();
```

`old_function`

Der Befehl `old_function` erlaubt es, ein Funktion genauso wie bei PHP/FI2 zu definieren (nur "function" wird durch "old_function" ersetzt.

Dieser Befehl sollte nur vom PHP/FI2->PHP3-Konverter genutzt werden.

Warnung

Funktionen, die durch `old_function` definiert werden, können nicht von PHP selbst aufgerufen werden. Das heißt unter anderem, dass sie nicht in Funktionen wie **usort ()**, array_walk() und **register_shutdown_function()** benutzt werden können. Dieses Manko können Sie umgehen, indem Sie eine Adapterfunktion in normalem PHP3 schreiben, die die `old_function` aufruft.

Variablenfunktionen

PHP unterstützt das Konzept der Variablenfunktionen. Wenn Sie an das Ende einer Variablen Klammern hängen, versucht PHP eine Funktion aufzurufen, deren Name der aktuelle Wert der Variable ist. Diese Möglichkeit kann für Callbacks, Funktionstabellen und

Beispiel 12-1. Beispiel für Variablenfunktionen

```
<?php
function blah() {
    echo "In blah()<br>\n";
}

function fasel($arg = ”) {
    echo "In fasel(); ist der Parameter ’$arg’.<br>\n";
}

$func = ’blah’;
$func();
$func = ’fasel’;
$func(’test’);
?>
```

Kapitel 13. Klassen und Objekte

Klassen

Eine Klasse ist eine Sammlung von Variablen und von Funktionen, die mit diesen Variablen arbeiten. Eine Klasse wird folgendermaßen definiert:

```
<?php
class Einkaufswagen {
    var $produkte;  // Produkte in Ihrem Einkaufswagen

    // Füge dem Einkaufswagen $anzahl Artikel der Sorte $artnr zu

    function fuege_hinzu ($artnr, $anzahl) {
        $this->produkte[$artnr] += $anzahl;
    }

    // Nimm $anzahl von Artikel wieder aus dem Einkaufswagen

    function nimm_heraus ($artnr, $anzahl) {
        if ($this->produkte[$artnr] > $anzahl) {
            $this->produkte[$artnr] -= $anzahl;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

In diesem Beispiel wird eine Klasse "Einkaufswagen" definiert. Sie besteht aus einem assoziativen Array von Produkten im Einkaufswagen und zwei Funktionen zum Zufügen und Entfernen von Einkäufen.

Klassen sind Typen, dass heisst die Blaupausen für realle Variablen. Um sie zu nutzen, muß zunächst eine Variable mit dem Operator `new` angelegt werden.

```
$einkaufswagen = new Einkaufswagen;
$einkaufswagen->fuege_hinzu("10", 1);
```

Hier wird das Objekt $einkaufswagen aus der Klasse Einkaufswagen geschaffen. Dann wird die enthaltene Funktion fuege_hinzu() aufgerufen, um ein Produkt mit der Artikelnummer 10 in den Einkaufswagen zu tun.

Klassen können ebenfalls Erweiterungen von anderen Klassen sein. Die erweiterte, oder auch abgeleitete Klasse enthält alle Funktionen der ursprünglichen Klasse, und dazu die eigenen Ergänzungen. Das geschieht mit dem Schlüsselwort "extends". Mehrfachvererbung wird von PHP nicht unterstützt.

```
class Mein_Einkaufswagen extends Einkaufswagen {
    var $besitzer;

    function setze_besitzer ($name) {
        $this->besitzer = $name;
    }
```

```
}
```

In diesem Beispiel wird eine Klasse Mein_Einkaufswagen definiert. Sie enthält alle Funktionen der Klasse Einkaufswagen, eine zusätzliche Variable $besitzer und die zusätzliche Funktion setze_besitzer(). Man kann den Einkaufswagen auch weiterhin wie oben erzeugen, nur kann man jetzt auch den Besitzer setzen oder herausfinden. Die alten Funktionen der Klasse Einkaufswagen können ebenfalls weiterverwendet werden.

```
$meinkaufswagen = new Mein_Einkaufswagen; // Kreiere einen Einkaufswagen
$meinkaufswagen->setze_besitzer ("kris"); // Name dieser Klasse
print $meinkaufswagen->besitzer;          // schreibe den Namen des Besitzers
$meinkaufswagen->fuege_hinzu ("10", 1);   // (Siehe oben, vererbt
                                          // von Einkaufswagen)
```

Innerhalb der Funktionen einer Klasse bezeichnet die Variable $this das aktuelle Objekt. Sie können mit $this->irgendwas auf dessen Variablen und Funktionen zugreifen.

Konstruktoren sind Funktionen eine Klasse, die beim Erschaffen eines neuen Objektes automatisch aufgerufen werden. Eine Funktion wird zu einem Konstruktor, wenn Sie den gleichen Namen wie die Klasse trägt.

```
class Auto_Einkaufswagen extends Einkaufswagen {
    function Auto_Einkaufswagen () {
        $this->fuege_hinzu ("10", 1);
    }
}
```

Die Klasse Auto_Einkaufswagen entspricht der Klasse Einkaufswagen plus einen Konstruktor, der bereits für eine erste Füllung (1 Artikel der Nummer 10) gesorgt hat. Jeder neu erzeugte Auto_Einkaufswagen enthält so von vorneherein diesen Artikel. Konstruktoren können auch Parameter enthalten. Aber diese Parameter sind optional, und können so nützlicher eingesetzt werden.

```
class Konstruktor_Einkaufswagen extends Einkaufswagen {
    function Konstruktor_Einkaufswagen ($produkt = "10", $anzahl = 1) {
        $this->fuege_hinzu ($produkt, $anzahl);
    }
}

// Kaufe wieder den gleichen alten Kram ein.

$standard_einkaufswagen   = new Konstruktor_Einkaufswagen;

// Kaufe etwas bestimmtes ein ...

$anderer_Einkaufswagen = new Konstruktor_Einkaufswagen ("20", 17);
```

Achtung

Bei abgeleiteten Klassen wird der Konstruktor der Ursprungsklasse nicht automatisch aufgerufen, wenn der Konstruktor der abgeleiteten Klasse aufgerufen wird.

Kapitel 14. Referenzen in PHP

Was Referenzen sind

Referenzen sind in PHP ein Mechanismus um verschiedene Namen für den gleichen Inhalt von Variablen zu ermöglichen. Sie sind nicht mit Zeigern in C zu vergleichen, sondern Aliasdefinitionen für die Symboltabelle. PHP unterscheidet zwischen Variablenname und Variableninhalt, wobei der gleiche Variableninhalt unterschiedliche Namen besitzen kann. Der bestmögliche Vergleich ist der mit Dateinamen und Dateien im Dateisystem von Unix - Variablennamen sind Verzeichniseinträge, während der Variableninhalt die eigentliche Datei darstellt. Referenzen können nun als Hardlinks im Dateisystem verstanden werden.

Was Referenzen leisten

PHP Referenzen erlauben es, zwei Variablennamen sich auf den gleichen Variableninhalt beziehen zu lassen. Das heisst im folgenden Beispiel, dass sich `$a` und `$b` auf dieselbe Variable beziehen:

```
$a =&amp; $b
```

Anmerkung: `$a` und `$b` sind hier gleichwertig, und `$a` ist nicht nur ein Zeiger auf `$b` oder umgekehrt, sondern `$a` und `$b` zeigen auf den selben Inhalt.

Seit PHP 4.0.4 kann `&` auch in Verbindung mit `new` verwendet werden.

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

Anmerkung: Wenn der `&` Operator nicht verwendet wird, erzeugt PHP eine Kopie des Objekts. Wenn nun `$this` innerhalb der Klasse verwendet wird, bezieht es sich auf die aktuelle Instanz der Klasse. Die Zuordnung ohne `&` erzeugt eine Kopie der Instanz (d.h. des Objekts) und `$this` wird sich auf die Kopie beziehen. In der Regel will man aus Performance- und Speicherverbrausgründen nur eine einzige Instanz einer Klasse erzeugen. Dafür stellt PHP den `&` bereit.

Eine weitere Einsatzmöglichkeit von Referenzen ist die Übergabe von Parametern an eine Funktion mit pass-by-reference. Hierbei beziehen sich der lokale Variablenname als auch der Variablenname der aufrufenden Instanz auf denselben Variableninhalt:

```
function foo (&amp;$var) {
    $var++;
```

```
}

$a=5;
foo ($a);
```

Nach der Ausführung hat `$a` den Wert 6, da sich in der Funktion `foo` der Variablenname `$var` auf denselben Variableninhalt bezieht wie `$a` in der aufrufenden Instanz (hier das Hauptprogramm).

Daneben besteht die Möglichkeit aus Funktionen heraus Werte mit return by-reference zurückzugeben.

Was Referenzen nicht sind

Wie bereits gesagt: Referenzen sind keine Zeiger. Das bedeutet, der folgende Code tut nicht, was zum Beispiel ein C Programmierer erwarten würde:

```
function foo (&amp;$var) {
    $var =&amp; $GLOBALS["baz"];
}
foo($bar);
```

Folgendes wird passieren: `$var` in foo wird zunächst an `$bar` aus der aufrufenden Instanz, dann aber an `$GLOBALS["baz"]`, gebunden. Es gibt keine Möglichkeit, `$bar` aus der aufrufenden Instanz mittels Referenz-Mechanismen an etwas anderes zu binden, da `$bar` in der Funktion foo nicht zur Verfügung steht (`$bar` wird durch `$var` repräsentiert; `$var` verfügt nur über Variableninhalt, besitzt aber keinen name-to-value Eintrag in der Symboltabelle der aufrufenden Instanz).

Referenzen zurückgeben

Das Zurückgeben von Ergebnissen per Referenz aus Funktionen heraus kann manchmal recht nüzlich sein. Hierbei ist folgende Syntax zu beachten:

```
function &amp;find_var ($param) {
    ...code...
    return $found_var;
}

$foo =&amp; find_var ($bar);
$foo->x = 2;
```

In diesem Beispiel wird also die Eigenschaft des von `find_var` gelieferten Objektes gesetzt, nicht die der Kopie, wie es der Fall wäre, wenn die Funktion `find_var` ihr Ergebnis nicht per Referenz liefern würde.

> **Anmerkung:** Im Gegensatz zur Parameterübergabe per Referenz ist bei der Rückgabe mittels Referenz an beiden Stellen die Angabe des `&` notwendig.

Referenzen aufheben

Wird eine Referenz aufgehoben, so wird nur die Bindung zwischen einem Variablennamen und dem Variableninhalt entfernt. Der Inhalt der Variablen wird hierbei nicht gelöscht. Betrachten wir folgendes Beispiel:

```
$a = 1;
$b =&amp; $a;
unset ($a);
```

Die Variable `$b` wird nicht gelöscht, sondern es wird nur die Bindung des Variablennamen `$a` an den Variableninhalt aufgehoben. Dieser Variableninhalt ist immer noch über `$b` verfügbar.

Wiederum sieht man die Analogie zwischen Unix und den Referenzen: Das Aufheben einer Referenz entspricht einem Aufruf von **unlink** unter Unix.

Referenzen entdecken

Viele Sprachelemente von PHP sind intern mit der Benutzung von Referenzen implementiert, daher gilt alles bisher gesagte auch für diese Konstrukte:

`global` References

Die Verwendung von **global $var** erzeugt im aktuellen Scope eine Referenz auf die globale Variable `$var`, sie ist also äquivalent zu folgendem:

```
$var =&amp; $GLOBALS["var"];
```

Dies hat zur Folge, dass das Anwenden von unset() auf `$var` keinen Einfluss auf die globale Variable hat.

`$this`

In einer Objektmethode ist `$this` immer eine Referenz auf die aufrufende Objektinstanz.

Teil III. Features

Kapitel 15. Fehlerbehandlung

Es gibt 4 verschiedene Arten von Fehlermeldungen und Warnungen in PHP. Das sind:

- 1 - Normale Funktionsfehler
- 2 - Normale Warnungen
- 4 - Parser-Fehler
- 8 - Mitteilungen (Warnungen, die ignoriert werden können, die aber eventuell auf einen Fehler im Code schließen lassen.)

Die oben aufgeführten Fehlernummern werden addiert, um den Fehlerlevel festzulegen. Der standardmäßige Wert ist 7, d.h. 1 + 2 + 4 oder, anders ausgedrückt, es werden alle Fehlermeldungen außer den Mitteilungen ausgegeben. Der Fehlerlevel kann in der Konfigurationsdatei php3.ini durch die Option error_reporting gesetzt werden. Es ist auch möglich, den Fehlerlevel in der Apache-Konfigurationsdatei httpd.conf durch die Option php3_error_reporting oder zur Laufzeit durch ein Skript, das die Funktion error_reporting() nutzt, zu setzen.

Alle PHP-Funktionen können auch mit vorangestelltem "@" aufgerufen werden. Das bewirkt das Unterdrücken der Fehlermeldung für die Funktion. Bei auftretendem Fehler und eingeschalteter track_errors Option wird die entsprechende Fehlermeldung in der globalen Variablen $php_errormsg gespeichert.

Kapitel 16. Erzeugen und Verändern von Bildern

PHP ist nicht darauf beschränkt, nur HTML auszugeben. Es kann auch dazu genutzt werden, Bilddateien in einer Vielzahl verschiedener Formate, darunter GIF, PNG, JPEG, WBMP und XPM, zu erzeugen und zu verändern. Noch angenehmer ist, dass PHP Bilddaten direkt an einen Browser ausgeben kann. Um all das zu tun, müssen sie nur die GD-Library mit Bildfunktionen eincompilieren. Es kann sein, daß PHP und GD noch weitere Bibliotheken benötigen, je nachdem, mit welchen Bildformaten Sie arbeiten möchten.

Beispiel 16-1. PNG-Erzeugung mit PHP

```
<?php
    Header("Content-type: image/png");
    $string=implode($argv," ");
    $im = imagecreatefrompng("images/taste1.png");
    $orange = ImageColorAllocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im,3,$px,9,$string,$orange);
    ImagePng($im);
    ImageDestroy($im);
?>
```

Dieses Beispiel würde von einer Seite mit einem Tag wie diesem aufgerufen: <img src="taste.php3?text">. Der obige Skript taste.php nimmt dann den String "text" und legt ihn über das Grundbild, in diesem Fall "images/taste1.png", und gibt das endgültige Bild aus. Das ist ein wirklich komfortabler Weg zu vermeiden, dass man jedesmal, wenn man den Text auf Tasten ändert, diese von Hand neu zeichnen muss, denn sie werden dynamisch generiert.

Kapitel 17. HTTP-Authentifizierung mit PHP

Die HTTP-Authentifizierung durch PHP ist nur verfügbar, wenn PHP als Apache-Modul läuft und funktioniert daher nicht mit der CGI-Version. In einem PHP-Skript für ein Apache-Modul kann man die Funktion header() benutzen, um die Nachricht "Authentifizierung notwendig" an den Client-Browser zu senden, damit dieser ein Fenster zur Eingabe von Benutzername/Passwort öffnet. Hat der Benutzer diese eingegeben, wird die URL des PHP-Scripts mit den Variablen $PHP_AUTH_USER, $PHP_AUTH_PW und $PHP_AUTH_TYPE, die den jeweiligen Benutzernamen, das Passwort und den Typ der Identifizierung enthalten, erneut aufgerufen. Momentan wird nur das Authentifizierungsmodell "basic" unterstützt. Näheres hierzu bei der header() Funktion.

Ein Auszug aus einem Skript, das die Clientauthentifizierung auf einer Seite erzwingt, würde so aussehen:

Beispiel 17-1. HTTP-Authentifizierung

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"My Realm\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Text to send if user hits Cancel button\n";
    exit;
  } else {
    echo "Hello $PHP_AUTH_USER.<P>";
    echo "You entered $PHP_AUTH_PW as your password.<P>";
  }
?>
```

Hinweis: Achten Sie bei den Header-Zeilen für maximale Kompatibilität auf die richtige Schreibweise! Das Schlüsselwort "Basic" sollte genau so geschrieben werden, der Realm-String muss in doppelte (nicht einfache) Anführungszeichen eingeschlossen sein, und in der "HTTP/1.0 401"-Zeile darf nur genau ein Leerzeichen vor dem 401-Code stehen.

Anstatt $PHP_AUTH_USER und $PHP_AUTH_PW einfach nur auszugeben, werden Sie den Benutzernamen und das Passwort auf Gültigkeit überprüfen wollen. Dies kann durch die Abfrage einer Datenbank oder das Einlesen einer Textdatei geschehen.

Vorsicht bei einigen Internet Explorer-Versionen - sie scheinen sehr wählerisch zu sein, was die Reihenfolge der Header angeht. Abhilfe schafft hier das Senden des *WWW-Authenticate* Headers vor dem HTTP/1.0 401.

Um zu unterbinden, dass ein Skript das Passwort einer durch einen traditionellen externen Mechanismus geschützten Seite ausliest, werden die PHP_AUTH Variablen nicht gesetzt, wenn eine externe Authentifizierung für diese bestimmte Seite aktiviert ist. In diesem Fall kann die $REMOTE_USER Variable benutzt werden, um den Benutzer durch die externe Zugriffskontrolle zu identifizieren.

Konfigurationshinweis: PHP prüft das Vorhandensein einer `AuthType` Apache-Direktive, um zu entscheiden, ob eine externe Authentifizierung aktiv ist. Vermeiden Sie deshalb diese Konfigurationsdirektive im Kontext der PHP-Authentifizierung (anderenfalls wird jeder Authentifizierungsversuch misslingen).

Zu beachten ist, dass obenstehendes keinesfalls jemanden, der die Kontrolle über eine nichtgeschützte URL hat, davon abhalten kann, Passwörter von geschützten URLs auf dem gleichen Rechner auszulesen.

Sowohl Netscape als auch der Internet Explorer löschen den lokalen Authentifizierungscache des Browserfensters, wenn der Server eine 401-Meldung zurückgibt. Dies kann benutzt werden, um einen Benutzer "auszuloggen" und eine erneute Eingabe des Benutzernamens/Passworts zu erzwingen. Manchmal wird dieses Verhalten für das automatische Ausloggen nach Ablauf einer bestimmten Zeitspanne oder für einen Logout-Button genutzt.

Beispiel 17-2. HTTP-Authentifizierung, mit erneuter Anforderung von Name/Passwort

```
<?php
  function authenticate() {
   Header( "WWW-Authenticate: Basic realm=\"Test Authentication System\"");
   Header( "HTTP/1.0 401 Unauthorized");
   echo "You must enter a valid login ID and password to access this resource\n";
   exit;
  }

  if(!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth, $PHP_AUTH_USER)) )
   authenticate();
  }
  else {
   echo "Welcome: $PHP_AUTH_USER<BR>";
   echo "Old: $OldAuth";
   echo "<FORM ACTION=\"$PHP_SELF\" METHOD=POST>\n";
   echo "<INPUT TYPE=HIDDEN NAME=\"SeenBefore\" VALUE=\"1\">\n";
   echo "<INPUT TYPE=HIDDEN NAME=\"OldAuth\" VALUE=\"$PHP_AUTH_USER\">\n";
   echo "<INPUT TYPE=Submit VALUE=\"Re Authenticate\">\n";
   echo "</FORM>\n";
  }
?>
```

Dieses Verhalten wird vom Authentifizierungsstandard HTTP Basic nicht gefordert, daher sollte man sich nie darauf verlassen. Tests mit Lynx haben gezeigt, dass Lynx die Authentifizierungsinformationen bei Erhalt einer 401-Meldung nicht löscht. Ein Klick auf den Zurück- Button und danach auf Vorwärts wird die angeforderte Adresse öffnen (und zwar so lange, bis die Identifizierung der Benutzer geändert wird).

Weiterhin muss beachtet werden, dass dies unter dem Microsoft IIS mit der CGI-Version von PHP aufgrund einer Einschränkung des IIS nicht funktioniert.

Kapitel 18. Cookies

PHP unterstützt HTTP-Cookies. Hierbei handelt es sich um einen Mechanismus, um Informationen beim Client zu speichern und somit wiederkehrende Besucher zu identifizieren oder ihren Weg innerhalb des Angebotes nachzuvollziehen. Cookies können durch die Funktion setcookie() gesetzt werden. Sie sind Bestandteil des HTTP-Headers, was bedeutet, daß die Funktion setcookie() aufgerufen werden muß, bevor irgendeine Ausgabe an den Browser erfolgt. Dies ist die gleiche Einschränkung, der auch die Funktion header() unterliegt.

Vom Client gesendete Cookies werden automatisch in eine Variable geschrieben, wie auch bei GET oder POST. Sollen einem Cookie mehrere Werte zugewiesen werden, muss dem Cookienamen lediglich *[]* angefügt werden. Einzelheiten siehe Funktion setcookie().

Kapitel 19. Steuerung von Dateiuploads

Dateiuploads mit POST

PHP kann Dateiuploads mit jedem RFC-1867 konformen Browser (dazu gehören der Natscape Navigator 3 oder höher, Microsoft Internet Explorer 3 mit Patch von Microsoft oder höher ohne Patch) durchführen. Es können sowohl Text- als auch Binärdaten hochgeladen werden. Mit PHP´s Authentifizierungs- und Dateifunktionen besteht volle Kontrolle darüber, wer Dateien hochladen darf und was mit den Dateien geschehen soll, wenn der Upload beendet ist.

PHP unterstützt auch Dateiuploads nach der PUT-Methode, die beispielsweise vom Netscape Composer und den W3C Amaya Clients benutzt wird. Siehe dazu PUT-Unterstützung für nähere Informationen.

Eine Maske für den Dateiuplod kann erstellt werden, indem man ein Formular entwirft, was ungefähr so aussieht:

Beispiel 19-1. Formular für den Dateiupload

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Die URL sollte auf eine PHP-Datei verweisen. Das versteckte Feld MAX_FILE_SIZE muß dem Dateieingabefeld vorausgehen und den Wert der maximal akzeptierten Dateigrösse in Bytes enthalten. Bei erfolgreichem Upload des Files werden folgende Variablen definiert:

- $userfile - Der temporäre Name, unter dem die hochgeladene Datei auf demServer gespeichert wurde.
- $userfile_name - Der ursprüngliche Dateiname auf dem System des Absenders.
- $userfile_size - Größe der hochgeladenen Datei in Bytes.
- $userfile_type - Der Mime-Typ der Datei, wenn dies Information zur Verfügung gestellt wird. Ein Beispiel wäre "image/gif".

Die Variable "$userfile" trägt den Namen des Dateifeldes im Formular; im obigen Beispiel haben wir die Bezeichnung "userfile" gewählt.

Standardmäßig werden Dateien in dem vorgegebenen temporären Verzeichnis des Servers gespeichert. Dies kann durch das Setzen der Umgebungsvariablen TMPDIR direkt in der Umgebung, in der PHP ausgeführt wird, geändert werden. Ein Setzen des temporären Verzeichnisses durch die Funktion putenv() innerhalb eines Skriptes ist nicht möglich.

Das PHP-Skript, dem die hochzuladende Datei übergeben wird, sollte alle nötigen Anweisungen enthalten, die festlegen, wie mit der hochgeladenen Datei verfahren werden soll. Beispielsweise kann die $file_size - Variable verwendet werden, um Dateien auszusortieren, die zu gross oder zu klein sind. Oder man benutzt die $file_type - Variable, um sich aller Dateien zu entledigen, die nicht bestimmten Typen entsprechen. Auf jeden Fall sollte die hochgeladene Datei aus dem temporären Verzeichnis gelöscht oder an andere Stelle verschoben werden.

Die Datei wird aus dem temporären Verzeichnis gelöscht, sobald das Skript abgearbeitet ist, wenn sie nicht verschoben oder umbenannt wurde.

Häufige Probleme

Der mit MAX_FILE_SIZE eingestellte Wert kann nicht grösser sein als der durch die Zeile upload_max_filesize der PHP3.ini (oder der entsprechenden Zeile php3_upload_max_filesize in der Apache Konfigurationsdatei) festgelegte. Standardmäßig sind dies 2 MB.

Bitte beachten, daß der CERN httpd alles abschneidet, was auf das erste Leerzeichen im content-type Mime-Header folgt, den er vom Client erhält. Solange dies der Fall ist, unterstützt der CERN httpd keine Dateiuploads.

Mehrere Dateien uploaden

Es ist möglich, gleichzeitig mehrere Dateien hochzuladen und die Upload-Informationen automatisch in Arrays aufgeführt zu erhalten. Dabei muß im HTML-Formular die gleiche Syntax zum Übermitteln des Arrays benutzt werden, was man durch verschiedene Select- und Checkboxen erreicht.

> **Anmerkung:** Die Unterstützung für den Upload mehrerer Dateien wurde in Version 3.0.10 hinzugefügt.

Beispiel 19-2. Formular zum Upload mehrerer Dateien

```
<form action="file-upload.html" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

Wenn das obige Formular abgeschickt wird, werden die Arrays `$userfile`,`$userfile_name` und `$userfile_size` für den globalen Bereich, wie in $HTTP_POST_VARS, angeordnet. Jede dieser Variablen ist ein numerisch indizierte Array, das aus den entsprechenden Werten der übertragenen Dateien besteht.

Nehmen wir an, die Dateinamen `/home/test/review.html` und `/home/test/xwp.out` werden übertragen. In diesem Fall beinhaltet`$userfile_name[0]` den Wert `review.html` und `$userfile_name[1]` enthält den Wert `xwp.out`. Genauso enthält `$userfile_size[0]` die Dateigröße von `review.html`, und so weiter.

PUT-Unterstützung

PHP unterstützt die HTTP PUT-Methode, wie sie von Clients wie Netscape Composer oder W3C Amaya benutzt wird. PUT-Anfragen sind weitaus unkomplizierter als ein Dateiupload und sehen etwa so aus:

```
PUT /path/filename.html HTTP/1.1
```

Das würde normalerweise bedeuten, daß der entfernte Client den folgenden Inhalt als /path/filename.html im Webverzeichnisbaum speichert. Natürlich ist es keine gute Idee, dass PHP oder Apache jeden Benutzer beliebige Dateien überschreiben lassen. Um eine solche Anfrage bearbeiten zu können, muss der Webserver erst angewiesen werden, ein bestimmtes PHP-Skript für die Abarbeitung aufzurufen. Im Apache wird dies durch die *Script* - Direktrive festgelegt. Sie kann irgendwo in der Apache-Konfigurationsdatei plaziert werden, gebräuchlich ist die Plazierung innerhalb einer <Directory>- oder <Virtualhost>- Sektion. Eine Zeile wie die folgende erledigt dies:

```
Script PUT /put.php3
```

Diese Zeile legt fest, daß Apache alle PUT-Anfragen für URIs, die dem Kontext entsprechen, in dem diese Zeile steht, an das put.php3 Skript weiterleitet. Dies setzt voraus, daß PHP aktiv und für die .php3-Dateierweiterung registriert ist.

Innerhalb der put.php3-Datei könnte folgendes stehen:

```
<? copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

Dies kopiert die Datei an den vom Client angegebenen Ort. Möglicherweise sollen vor dem Kopieren der Datei noch Überprüfungen und/oder Benutzerauthentifizierung stattfinden. Wenn PHP eine PUT-Anfrage erhält, wird die Datei in einem temporären Verzeichnis gespeichert, genau wie bei POST-Methode Nach Abarbeiten der Anfrage wird die temporäre Datei gelöscht. Also muß das PHP-Skript, das die PUT-Anfrage bearbeitet, die Datei irgendwohin kopieren. Der Name der temporären Datei findet sich in der Variablen $PHP_PUT_FILENAME; der Name der zu speichernden Datei steht in $REQUEST_URI (kann bei Nicht-Apache Webservern variieren). Dieser Zieldateiname wird vom Client festgelegt. Man kann den Client auch umgehen; beispielsweise könnten alle hochgeladenen Dateien in ein spezielles Upload-Directory kopiert werden.

Kapitel 20. Zugriff auf entfernte Dateien

Wenn die Unterstützung für den "URL fopen wrapper" bei der Konfiguration von PHP vorhanden ist (welche standardmäßig eingestellt ist, es sei denn, die Option `--disable-url-fopen-wrapper` wurde bei der Installation verwendet (Versionen bis 4.0.3) oder der Parameter *`allow_url_fopen`* in der php.ini deaktiviert wurde (neuere Versionen), können HTTP und FTP URLs bei den meisten Funktionen verwendet werden, die einen Dateinamen als Parameter benötigen, inklusive require() und include() Anweisungen.

> **Anmerkung:** remote files funktionieren nicht mit include() und require() Anweisungen unter Windows.

Beispielsweise kann eine Datei auf einem anderen Webserver geöffnet und verarbeitet werden. Diese Daten können zur Abfrage einer Datenbank benutzt werden oder passend zum Rest der eigenen Website ausgegeben werden.

Beispiel 20-1. Den Titel einer entfernten Seite auslesen

```
<?php
$file = fopen ("http://www.php.net/", "r");
if (!$file) {
    echo "<p>Datei konnte nicht ge&uuml;ffnet werden.\n";
    exit;
}
while (!feof ($file)) {
    $line = fgets ($file, 1024);
    /* Funktioniert nur, wenn Titel und title-Tags in einer Zeile stehen */
    if (eregi ("<title>(.*)</title>", $line, $out)) {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

Auch eine Datei auf einem FTP-Server kann geschrieben werden, solange man sich über einen Benutzer mit entsprechenden Zugriffsrechten verbindet und die Datei noch nicht existiert. Um sich mit einem anderen Benutzer als ’anonymous’ zu verbinden muß ein username (und möglichst ein Passwort) innerhalb der URL angegeben werden, wie z.B. ’ftp://user:password@ftp.example.com/pfad/zur/datei’. (Die selbe Syntax kann verwendet werden, um auf Daten via HTTP zuzugreifen, wenn diese eine Basic Authentication benötigen.)

Beispiel 20-2. Daten auf einen entfernten Server speichern

```
<?php
$file = fopen ("ftp://ftp.php.net/incoming/outputfile", "w");
if (!$file) {
```

```
    echo "<p>Datei konnte zum schreiben nicht ge&ouml;ffnet werden.\n";
    exit;
}
/* Schreibe die Daten hier hin. */
fputs ($file, "$HTTP_USER_AGENT\n");
fclose ($file);
?>
```

Anmerkung: Obiges Beispiel könnte dazu verleiten, dieses Verfahren zu benutzen, um in ein 'remote log-file' zu schreiben. Wie oben erwähnt kann man jedoch ausschließlich neue Dateien anlegen, wenn man URL fopen() wrapper benutzt. Für 'distributed logging' sollte man sich die Funktion syslog() anschauen.

Kapitel 21. Verbindungssteuerung

Anmerkung: Folgendes trifft auf die Versionen 3.0.7 und später zu.

PHP erhält intern einen Verbindungsstatus. Dieser kann drei Zustände annehmen:

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

Wenn ein PHP-Skript aktiv ist, ist der Status üblicherweise NORMAL. Sollte der Client-Rechner die Verbindung beenden, wird der Status auf ABORTED gesetzt. Ein clientseitiges Beenden der Verbindung wird für gewöhnlich veranlaßt, wenn der Benutzer den STOP-Button seines Browsers drückt. Wenn das eingestellte Zeitlimit (siehe set_time_limit()) überschritten wird, wird der Status TIMEOUT gesetzt.

Sie können entscheiden, ob der Verbindungsabbruch seitens des Clients den Abbruch des Skriptes zur Folge haben soll. Manchmal ist es sinnvoll, Skripte sauber zu beenden, auch wenn kein Browser mehr zur Verfügung steht, der die Ausgabe empfängt. Die Abarbeitung eines Skriptes wird standardmäßig abgebrochen, wenn der Client die Verbindung beendet. Dieses Verhalten kann sowohl durch die Option ignore_user_abort in der Konfigurationsdatei php3.ini, durch die entsprechende Option php3_ignore_user_abort in der Apache-Konfigurationsdatei als auch durch ignore_user_abort() beeinflußt werden. Wenn PHP nicht angewiesen wird, einen Verbindungsabbruch durch den Benutzer zu ignorieren und die Verbindung dann durch den Benutzer beendet wird, wird die Abarbeitung des Scriptes abgebrochen. Die einzige Ausnahme ist, wenn durch die Funktion register_shutdown_function() eine Shutdown-Funktion angegeben wird, die bei clientseitigem Abbruch ausgeführt wird. Wenn dann der Benutzer den STOP-Button seines Browsers drückt, wird PHP bei der nächsten Ausgabe des Skriptes feststellen, daß die Verbindung abgebrochen wurde und die Shutdown-Funktion aufrufen. Diese Shutdown-Funktion wird auch aufgerufen, wenn das Skript auf normalem Wege beendet wird, daher sollte man, wenn man für den Fall eines Benutzerabbruchs etwas anderes vorgesehen hat, die Funktion connection_aborted() verwenden. Sie gibt `TRUE` zurück, wenn die Verbindung abgebrochen wurde.

Ein Skript kann ebenfalls durch den eingebauten Script-Timer beendet werden. Der Standard-Timeout beträgt 30 Sekunden. Er kann durch die Option max_execution_time in der php3.ini, durch den entsprechenden Eintrag php3_max_execution_time in der Apache-Konfigurationsdatei oder durch die Funktion set_time_limit() beeinflußt werden. Bei Zeitüberschreitung wird das Skript beendet und, genau wie im obigen Fall des Verbindungsabbruchs, eine registrierte Shutdown-Funktion ausgeführt. Um zu überprüfen, ob es sich um einen Abbruch aufgrund von Zeitüberschreitung handelt, kann die Funktion connection_timeout() benutzt werden. Sie gibt `TRUE` zurück, wenn es sich um eine Zeitüberschreitung handelt.

Zu bemerken ist, daß der ABORTED und der TIMEOUT-Status gleichzeitig auftreten können. Dies ist möglich, wenn PHP angewiesen wird, Benutzerabbrüche zu ignorieren. PHP wird feststellen, daß der Benutzer die Verbindung abgebrochen hat, das Skript allerdings läuft weiter. Sollte es dann das Zeitlimit erreichen, wird es abgebrochen und eine Shutdown-Funktion, wenn definiert, wird aufgerufen. Zu diesem Zeitpunkt kann man feststellen, dass connection_timeout() und connection_aborted() `TRUE` zurückgeben. Diese beiden Statusmöglichkeiten können auch durch einen Aufruf der Funktion connection_status() abgefragt werden. Sie liefert ein Bitfeld des aktiven Status. Wenn beispielsweise TIMEOUT und ABORTED aktiv sind, wird 3 zurückgegeben.

Kapitel 22. Persistente Datenbankverbindungen

Persistente Verbindungen sind SQL-Verbindungen, die nach Abarbeitung des Skriptes nicht geschlossen werden. Wenn eine persistente Verbindung angefordert wird, prüft PHP zuerst, ob bereits eine identische persistente Verbindung (die vielleicht vorher offen geblieben ist) existiert und benutzt sie in diesem Fall. Sollte keine Verbindung existieren, wird eine hergestellt. Eine 'identische' Verbindung ist eine Verbindung, die zu dem gleichen Host mit dem gleichen Usernamen und Passwort hergestellt wurde.

Wer nicht durchgängig mit der Art und Weise, wie Webserver arbeiten und die Last verteilen, vertraut ist, könnte missverstehen, wofür persistente Verbindungen gedacht sind. Im Besonderen bieten sie *keine* Möglichkeit, 'Benutzersitzungen' auf der gleichen SQL-Verbindung zu öffnen und *keine* Möglichkeit, eine Transaktion aufzubauen, und sie können viele andere Sachen nicht. Um absulute Klarheit zu schaffen: Persistente Verbindungen bieten *keine* Funktionalität, die nicht auch von nicht-persistenten Verbindungen bereitgestellt wird.

Warum?

Das hat mit der Arbeitsweise von Webservern zu tun. Es gibt drei Möglichkeiten, wie ein Webserver PHP zur Generierung von Webseiten einsetzen kann.

Die erste Methode ist, PHP als CGI-'Wrapper' zu benutzen. Wenn diese Methode eingesetzt wird, wird für jede Anfrage nach einer PHP-Seite vom Webserver eine Instanz des PHP- Interpreters gestartet und anschliessend wieder beendet. Durch die Beendigung des Interpreters nach abgeschlossener Anfrage werden alle Ressourcen, auf die zugegriffen wurde (wie beispielsweise eine Verbindung zu einem SQL- Datenbankserver) wieder geschlossen. In diesem Fall erreicht man nichts, wenn man persistente Verbindungen benutzt - sie sind eben nicht beständig.

Die zweite und populärere Methode ist der Einsatz von PHP als Modul in einem Multiprozess-Webserver, was momentan nur auf den Apache zutrifft. Typischerweise hat ein Multiprozess- Webserver einen Prozess (den 'Eltern' Prozess), der einen Satz weiterer Prozesse (die 'Kinder') koordiniert, die die eigentliche Arbeit des Bereitstellens der Seiten übenehmen. Jede Anfrage, die von einen Client erfolgt, wird an einen untergeordneten Prozess, der noch keine andere Anfrage bearbeitet, weitergereicht. Das bedeutet, dass eine zweite Anfrage des Clients an den Server unter Umständen von einem anderen untergeordneten Prozess als die erste Anfrage bearbeitet wird. In diesem Fall sorgt eine persistente Verbindung dafür, dass jeder untergeordnete Prozess sich nur einmal mit dem SQL-Server verbinden muss, wenn eine solche benötigt wird. Benötigt dann eine weitere Seite die Verbindung mit dem SQL-Server, kann auf die zurückgegriffen werden, die der untergeordnete Prozess vorher hergestellt hat.

Die letzte Methode ist, PHP als Plug-in für einen Multithread- Webserver zu benutzen. Momentan ist dies noch Theorie - PHP arbeitet noch nicht als Plug-in für irgedeinen dieser Server. An der Unterstützung für ISAPI, WSAPI und NSAPI wird gearbeitet, so dass die Nutzung von PHP mit Multithread-Serven wie Netscape Fast Track, Microsoft Internet Information Server (IIS) und O'Reilly's WebSite Pro möglich wird. Wenn es soweit ist, wird das Verhalten im wesentlichen dem oben beschriebenen Multiprozess-Modell entsprechen.

Wozu dienen persistente Verbindungen, wenn sie keine zusätzliche Funktionalität bieten?

Die Antwort ist ausserordentlich einfach: Effizienz. Persistente Verbindungen sind nützlich, wenn die Notwendigkeit, eine Verbindung zu einem SQL-Server herzustellen, hoch ist. Ob dies der Fall ist oder nicht, hängt von vielen Faktoren ab - zum Beispiel, um was für eine Datenbank es sich handelt, ob sie auf dem gleichen Rechner wie der Webserver läuft oder welche Last die SQL-Maschine zu bewältigen hat usw. Grundsätzlich gilt, dass, wenn viele Verbindungen hergestellt werden müssen, persistente Verbindungen ausserordentlich hilfreich sind. Sie veranlassen den untergeordneten Prozess, sich während seiner gesamten Lebensdauer lediglich einmal mit dem SQL-Server zu verbinden, anstatt jedesmal beim Aufruf einer Seite, die eine Verbindung benötigt. Das heisst, dass jeder untergeordneten Prozess, der eine persistente Verbindung öffnet, eine eigene dauerhafte

Verbindung zum Server hat. Bei 20 untergeordnete Prozessen, die ein Skript ausführen, das eine persistente Verbindung zum SQL-Server herstellt, hat man beispielsweise 20 verschiedene Verbindungen zum SQL-Server - eine für jeden untergeordneten Prozess.

Eine wichtige Zusammenfassung. Persistente Verbindungen wurden entwickelt, um eins-zu-eins Abbildungen auf reguläre Verbindungen zu haben. Das heisst, dass man *immer* in der Lage sein sollte, die persistenten Verbindungen durch nicht-persistente zu ersetzten, ohne dass dies den Skriptablauf verändert. Es *kann* (und wird vermutlich auch) die Effizienz des Skriptes beeinflussen, aber nicht dessen Verhalten.

Kapitel 23. Safe Mode

Safe Mode ist der Versuch, Sicherheitsprobleme bei gemeinsam genutzten Servern zu lösen. Bezogen auf die Systemarchitektur, ist es der falsche Ansatz, diese Probleme innerhalb der PHP Schicht lösen zu wollen. Da es auf Ebene des Webservers bzw. des Betriebssystems keine praktischen Alternativen gibt, wird Safe Mode nunmehr von vielen Leuten, vor allem von Providern, eingesetzt.

Es existieren folgende Direktiven um Safe Mode zu konfigurieren:

```
safe_mode = Off
open_basedir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_functions =
```

Wenn safe_mode aktiviert ist, überprüft PHP, ob der Eigentümer des laufenden Skriptes dem Eigentümer der Datei entspricht, an der mit Hilfe der Dateifunktionen gearbeitet werden soll. Beispiel:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Bei der Ausführung dieses Skriptes

```
<?php
 readfile('/etc/passwd');
?>
```

erscheint bei aktiviertem Safe Mode folgende Warnung:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

Wenn anstelle von safe_mode ein open_basedir Verzeichnis angegeben wurde, können Dateioperationen nur noch unterhalb dieses Verzeichnisses vorgenommen werden. Beispiel (Apache httpd.conf):

```
<Directory /docroot>
php_admin_value open_basedir /docroot
</Directory>
```

Wenn das Skript script.php mit der open_basedir Einstellung ausgeführt wird, erscheint folgende Warnung:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

Darüberhinaus können einzelne Funktionen deaktiviert werden. Würde man folgenden Eintrag in der php.ini vornehmen:

```
disable_functions readfile,system
```

Würde diese Warnung erscheinen:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

Funktionen, die durch Safe Mode deaktiviert oder eingeschränkt werden

Dies ist eine Auflistung der Funktionen, die durch Safe Mode eingeschränkt werden. Diese Liste ist möglicherweise unvollständig und fehlerhaft!

Tabelle 23-1. Durch Safe Mode eingeschränkte Funktionen

Funktion	Einschränkungen
dbmopen()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
dbase_open()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
filepro()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
filepro_rowcount()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.

Funktion	Einschränkungen
filepro_retrieve()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
ifx_*()	sql_safe_mode restrictions, (!= Safe Mode)
ingres_*()	sql_safe_mode restrictions, (!= Safe Mode)
mysql_*()	sql_safe_mode restrictions, (!= Safe Mode)
pg_loimport()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
posix_mkfifo()	Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.
putenv()	Führt die safe_mode_protected_env_vars und safe_mode_allowed_env_vars ini-directiven aus. Siehe auch Dokumentation zu putenv()
move_uploaded_file()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
chdir()	Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.
dl()	Diese Funktion ist bei Verwendung von Safe Mode deaktiviert.
backtick operator	Diese Funktion ist bei Verwendung von Safe Mode deaktiviert.
shell_exec() (funktional gleichwertig zu den Backtick-Operatoren)	Diese Funktion ist bei Verwendung von Safe Mode deaktiviert.
exec()	Ausführbare Dateien können nur innerhalb des durch safe_mode_exec_dir angegebenen Verzeichnisses ausgeführt werden. Aus praktischen Gründen ist es momentan nicht erlaubt, `..` im Pfad zu den ausführbaren Datein anzugeben.
system()	Ausführbare Dateien können nur innerhalb des durch safe_mode_exec_dir angegebenen Verzeichnisses ausgeführt werden. Aus praktischen Gründen ist es momentan nicht erlaubt, `..` im Pfad zu den ausführbaren Datein anzugeben.
passthru()	Ausführbare Dateien können nur innerhalb des durch safe_mode_exec_dir angegebenen Verzeichnisses ausgeführt werden. Aus praktischen Gründen ist es momentan nicht erlaubt, `..` im Pfad zu den ausführbaren Datein anzugeben.

Funktion	Einschränkungen
popen()	Ausführbare Dateien können nur innerhalb des durch safe_mode_exec_dir angegebenen Verzeichnisses ausgeführt werden. Aus praktischen Gründen ist es momentan nicht erlaubt, `..` im Pfad zu den ausführbaren Datein anzugeben.
mkdir()	Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.
rmdir()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
rename()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.
unlink()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.
copy()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst. (on *`source`* and *`target`*)
chgrp()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
chown()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst.
chmod()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. In addition, you cannot set the SUID, SGID and sticky bits
touch()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.

Funktion	Einschränkungen
symlink()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst. (note: only the target is checked)
link()	Überprüft ob die Dateien/Verzeichnisse, die mit dem Skript bearbeitet werden sollen, die gleiche UID haben wie das Skript selbst. Überprüft ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst. (note: only the target is checked)
getallheaders()	Header die mit 'authorization' (unabhängig von Groß- oder Kleinschreibung) beginnen, werden bei Verwendung von Safe Mode nicht zurückgegeben. Achtung: Dies ist fehlerhaft bei der AOL-Server Implementation der Funktion getallheaders()!
Jede Funktion, die `php4/main/fopen_wrappers.c` benutzt...	??

Teil IV. Funktionsreferenz

I. Apache-spezifische Funktionen

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Führt eine Teilanfrage für eine URI durch und liefert alle Informationen darüber zurück

object **apache_lookup_uri** (string Dateiname)

apache_lookup_uri() führt eine Teilanfrage über eine bestimmte URI durch. Die Abfrage geht gerade soweit, dass alle wichtigen Informationen über die gegebene Quelle gesammelt werden können. Als Resultat wird eine Klasse zurückgeliefert, die folgende Elemente enthält:
status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

Anmerkung: apache_lookup_uri() funktioniert nur, wenn PHP als Apache-Modul eingerichtet wurde.

apache_note (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Setzt und liest Apache Notes

string **apache_note** (string note_name [, string note_value])

apache_note() ist eine Apache-spezifische Funktion, die Einträge in Apaches `notes`-Tabelle setzt und ausliest. Wenn sie mit einem einzelnen Parameter aufgerufen wird, gibt sie den aktuellen Wert der Note *`note_name`* zurück. Bei zwei Parametern wird der Eintrag *`note_name`* auf den zweiten Parameter *`note_value`* gesetzt und liefert den bisherigen Wert zurück.

ascii2ebcdic (PHP 3>= 3.0.17)

Zeichenkette von ASCII in EBCDIC übersetzen

int **ascii2ebcdic** (string ascii_str)

ascii2ebcdic() ist eine Apache-spezifische Funktion, die nur auf Systemen mit EBCDIC-Zeichensatz existiert (Betriebssysteme: OS/390, BS2000). Die Funktion setzt die ASCII-kodierte Zeichenkette *ascii_str* in ihre äquivalente EBCDIC-Darstellung um (binärfest), und liefert das konvertierte Ergebnis zurück.

Siehe auch die Umkehrfunktion ebcdic2ascii() .

ebcdic2ascii (PHP 3>= 3.0.17)

Zeichenkette von EBCDIC in ASCII übersetzen

int **ebcdic2ascii** (string ebcdic_str)

ebcdic2ascii() ist eine Apache-spezifische Funktion, die nur auf Systemen mit EBCDIC-Zeichensatz existiert (Betriebssysteme: OS/390, BS2000). Die Funktion setzt die EBCDIC-kodierte Zeichenkette *ebcdic_str* in ihre äquivalente ASCII-Darstellung um (binärfest), und liefert das konvertierte Ergebnis zurück.

Siehe auch die Umkehrfunktion ascii2ebcdic().

getallheaders (PHP 3, PHP 4 >= 4.0.0)

Liefert alle HTTP-Header der aktuellen Anfrage aus

array **getallheaders** (void)

Diese Funktion liefert ein assoziatives Array zurück, das alle HTTP-Header-Informationen der aktuellen HTTP-Anfrage enthält.

> **Anmerkung:** Die Werte der normalen CGI-Variablen können Sie auch durch das Auslesen der Umgebungsvariablen erhalten. Dieser Weg funktioniert unabhängig davon, ob Sie PHP als Apachemodul benutzen. Probieren Sie phpinfo() aus, um alle Umgebungsvariablen, die auf diese Weise erreichbar sind, zu erhalten.

Beispiel 1. getallheaders() Beispiel

```
$headers= getallheaders();
while (list ($header, $value) = each ($headers)) {
    echo "$header: $value&lt;br&gt;\n";
}
```

Dieses Beispiel gibt alle HTTP-Header der aktuellen Anfrage aus.

> **Anmerkung: getallheaders()** wird zur Zeit nur durch PHP als Apachemodul unterstützt.

virtual (PHP 3, PHP 4 >= 4.0.0)

Führt eine Apache-Unteranfrage durch

int **virtual** (string filename)

virtual() ist eine Apache-spezifische Funktion, die <!--#include virtual... --> aus mod_include entspricht. Sie führt eine Unteranfrage durch, die für das Einbinden von CGI-Skripten, .shtml-Dateien oder anderes, das vom Apache verarbeitet werden soll, nützlich ist. Beachten Sie bitte, dass ein CGI-Skript gültige CGI-Header generieren muss, mindestens einen Content-type-Header. Für PHP-Dateien müssen Sie include() oder require() benutzen, **virtual()** kann selbst keine PHP-Dokumente einbinden.

apache_child_terminate (PHP 4 >= 4.0.5)

Beendet einen Apache Prozess nach der Anfrage

string **apache_child_terminate** (void)

apache_child_terminate() beendet den Apache Prozess, der die aktuelle PHP-Anfrage ausführt, nachdem diese vollständig ausgeführt wurde. Sie können diese Funktion benutzen, nachdem Sie ein Skript mit hohem Speicherverbrauch ausgeführt haben. Der verbrauchte Speicher wird gewöhnlich nur intern, aber nicht für das Betriebsystem, freigegeben.

Siehe auch exit().

apache_setenv (PHP 4 CVS only)

Setzt eine Apache subprocess_env Variable

int **apache_setenv** (string Variable, string Wert [, bool walk_to_top])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

II. Array Funktionen

Diese Funktionen erlauben es, Arrays auf verschiedene Arten zu manipulieren bzw. mit ihnen zu interagieren. Arrays sind wichtig, um Gruppen von Variablen zu verwalten, zu bearbeiten, oder zu speichern.

Es werden einfache und mehrdimensionale Arrays unterstützt, welche entweder vom Benutzer oder von einer anderen Funktion erstellt werden können. Es gibt bestimmte Datenbankfunktionen, welche die Ergebnisse aus Datenbankabfragen in Arrays speichern, und verschiedene andere Funktionen, die Arrays als Rückgabewerte haben.

In der Sektion Arrays finden Sie eine detaillierte Erklärung, wie Arrays in PHP implementiert sind und wie sie benutzt werden.

Siehe auch is_array(), explode(), implode(), split() und join().

array (unknown)

Erstellt ein Array

array **array** ([mixed ...])

Gibt ein den Parametern entsprechendes Array zurück. Mit dem => Operator können die Parameter indiziert werden.

> **Anmerkung: array()** ist keine richtige Funktion sondern ein Sprachkonstrukt, das zur Erzeugung eines Wertefeldes genutzt wird.

Die Syntax "Index => Werte", durch Kommas getrennt, definiert Index und Werte. Index kann vom Typ String oder numerisch sein. Wird der Index weggelassen, erstellt die Funktion automatisch einen numerischen Index, der bei 0 beginnt. Ist der Index als Integer-Wert angegeben, wird der nächste generierte Index der größte Integer Index + 1. Beachten Sie, dass wenn zwei identische Indexe definiert sind, der letzte den ersten überschreibt.

Das folgende Beispiel zeigt wie man ein zweidimensionales Array erstellt, wie man Schlüssel für assoziative Arrays festlegt, und wie man numerische Indizes in normalen Arrays überspringt und fortsetzt.

Beispiel 1. array()

```
$fruits = array (
    "Früchte" => array ("a"=>"Orange", "b"=>"Banane", "c"=>"Apfel"),
    "Zahlen"  => array (1, 2, 3, 4, 5, 6),
    "Löcher"  => array ("erstes", 5 => "zweites", "drittes")
);
```

Beispiel 2. Automatischer Index mit array()

```
$array = array( 1, 1, 1, 1, 1, 8=>1, 4=>1, 19, 3=>13);
print_r($array);
```

wird folgendes anzeigen:

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
```

```
    [9] => 19
)
```

Beachten Sie, dass Index '3' doppelt definiert ist, und den letzten definierten Wert 13 behält. Index 4 wurde nach dem Index 8 definiert, und der nächste generierte Index (Wert 19) ist 9, da der größte Index 8 war.

Dieses Beispiel erstellt ein auf dem Index 1 basierendes Array.

Beispiel 3. 1-basierter Index mit array()

```
$erstesquartal  = array(1 => 'Januar', 'Februar', 'März');
print_r($erstesquartal);
```

wird folgendes ausgeben:

```
Array
(
    [1] => 'Januar'
    [2] => 'Februar'
    [3] => 'März'
)
```

Siehe auch array_pad(), list() und range().

array_change_key_case (PHP 4 CVS only)

Liefert ein Array mit allen String-Schlüsseln in Klein- oder Großbuchstaben

array **array_change_key_case** (array input [, int case])

array_change_key_case() wandelt die Schlüssel des Arrays *input* in Klein- bzw. Großbuchstaben um. Die Richtung der Umwandlung wird mit dem optionalen Parameter *case* festgelegt, mit dem die zwei Konstanten `CASE_UPPER` und `CASE_LOWER` übergeben werden können. Default ist `CASE_LOWER`. Diese Funktion lässt die numerischen Indizes unverändert.

Beispiel 1. array_change_key_case()

```
$input_array = array("FirSt" => 1, "SecOnd" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER);
```

wird folgendes ausgeben:

```
Array
(
    [FIRST] => 1
    [SECOND] => 2
)
```

array_chunk (PHP 4 CVS only)

Splittet ein Array in Teile auf

array **array_chunk** (array input, int size [, bool preserve_keys])

array_chunk() teilt das Array in verschiedene Arrays mit *size* Werten auf. Am Ende kann auch ein Array mit weniger Werten erzeugt werden. Die Arrays werden als Teile eines mehrdimensionalen Arrays erzeugt, welches bei Null und beginnend numerisch indiziert ist.

Sie können PHP dazu zwingen, die originalen Schlüssel des Arrays input beizubehalten, indem Sie den optionalen Parameter *preserve_keys* auf `TRUE` setzen. Geben Sie `FALSE` an, werden in jedem erzeugten Array neue numerische Indizes erzeugt, welche bei Null beginnen. Default ist `FALSE`.

Beispiel 1. array_chunk()

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

wird folgendes ausgeben:

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
```

```
        )

)
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

array_count_values (PHP 4 >= 4.0.0)

Zählt die Werte eines Arrays

array **array_count_values** (array input)

array_count_values() gibt ein Array zurück, in dem die Werte des Arrays *input* als Schlüssel, und deren Anzahl als Werte angegeben sind.

Beispiel 1. array_count_values()

```
$array = array (1, "hello", 1, "world", "hello");
print_r(array_count_values ($array));
```

Wird folgendes ausgeben:

```
Array
(
    [1] => 2
    [hello] => 2
    [world] => 1
)
```

array_diff (PHP 4)

Ermittelt die Unterschiede von Arrays

array **array_diff** (array array1, array array2 [, array ...])

array_diff() gibt ein Array mit allen Werten von *`array1`*, welche in keinem der anderen Argumente enthalten sind, zurück. Beachten Sie, dass Schlüssel erhalten bleiben.

Beispiel 1. array_diff()

```
$array1 = array ("a" => "grün", "rot", "blau", "rot");
$array2 = array ("b" => "grün", "gelb", "rot");
$result = array_diff ($array1, $array2);
```

Dies speichert `array("blau");` in `$result`. Mehrfache Vorkommen in $array1 werden auf die gleiche Weise behandelt.

Anmerkung: Zwei Elemente werden nur dann als gleich angesehen, wenn `(string) $elem1 === (string) $elem2`. In Worten: Wenn die String-Repräsentation die gleiche ist.

Warnung

Dies war in PHP 4.0.4 fehlerhaft!

Siehe auch array_intersect().

array_filter (PHP 4 >= 4.0.6)

Filtert Elemente eines Arrays mittels einer Callback-Funktion

array **array_filter** (array input [, mixed callback])

array_filter() gibt ein Array mit den mittels der Callback-Funktion gefilterten Elementen von *`input`* zurück. Ist *`input`* ein assoziatives Array, bleiben die Schlüssel erhalten.

Beispiel 1. array_filter()

```
function ungerade($var) {
    return ($var % 2 == 1);
}

function gerade($var) {
    return ($var % 2 == 0);
}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);

echo "Ungerade :\n";
print_r(array_filter($array1, "ungerade"));
echo "Gerade:\n";
print_r(array_filter($array2, "gerade"));
```

Wird folgendes ausgeben:

```
Ungerade :
Array
(
    [a] => 1
    [c] => 3
    [e] => 5
)
Gerade:
Array
(
    [0] => 6
    [2] => 8
    [4] => 10
    [6] => 12
)
```

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

Siehe auch array_map() und array_reduce().

array_flip (PHP 4 >= 4.0.0)

Vertauscht Werte und Schlüssel in einem Array

array **array_flip** (array trans)

array_flip() Gibt ein Array in umgekehrter Reihenfolge zurück, d.h. Schlüssel von `trans` werden Werte, und umgekehrt werden die Werte zu Schlüsseln.

Beachten Sie, dass die Werte von `trans` gültige Schlüssel sein müssen, z.B. haben sie entweder integer oder string zu sein. Ist ein Wert vom falschen Typ wird eine Warnung ausgegeben, und das Schlüssel/Werte-Paar *wird nicht vertauscht*.

Kommt ein Wert mehrmals vor, wird der letzte Schlüssel mit seinem Wert vertauscht, und alle anderen Schlüssel- Wertepaare gehen verloren.

array_flip() gibt im Fehlerfall `FALSE` zurück.

Beispiel 1. array_flip() Beispiel

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

Beispiel 2. array_flip() Beispiel: Kollision

```
$trans = array ("a" => 1, "b" => 1, "c" => 2);
$trans = array_flip ($trans);
print_r($trans);
```

Nun ist $trans:

```
Array
(
    [1] => b
    [2] => c
)
```

array_fill (PHP 4 CVS only)

Füllt ein Array mit Werten

array **array_fill** (int start_index, int num, mixed value)

array_fill() füllt ein Array mit *num* Einträgen des Wertes des *value* Parameters. Die Indizes des Arrays beginnen mit dem *start_index* Parameter.

Beispiel 1. array_fill()

```
$a = array_fill(5, 6, 'Banane');
```

$a würde mit print_r() folgende Inhalte ausgeben:

```
Array
(
    [5]  => Banane
    [6]  => Banane
    [7]  => Banane
    [8]  => Banane
    [9]  => Banane
    [10] => Banane
)
```

array_intersect (PHP 4)

Ermittelt die Schnittmenge von Arrays

array **array_intersect** (array array1, array array2 [, array ...])

array_intersect() gibt ein Array mit allen Werten von *array1*, die auch in allen anderen Argumenten enthalten sind, zurück. Beachten Sie, dass die Schlüssel erhalten bleiben.

Beispiel 1. array_intersect() Beispiel

```
$array1 = array ("a" => "grün", "rot", "blau");
$array2 = array ("b" => "grün", "gelb", "rot");
$result = array_intersect ($array1, $array2);
```

Danach enthält `$result`

```
Array
(
    [a] => grün
    [0] => rot
)
```

Anmerkung: Zwei Elemente werden nur dann als gleich angesehen, wenn `(string) $elem1 === (string) $elem2`. In Worten: Wenn die String-Repräsentation die gleiche ist.

Warnung

Dies war in PHP 4.0.4 fehlerhaft!

Siehe auch array_diff().

array_key_exists (PHP 4 >= 4.1.0)

Prüft, ob ein Schlüssel in einem Array existiert

bool **array_key_exists** (mixed key, array search)

array_key_exists() gibt `TRUE` zurück, wenn *`key`* in dem Array vorhanden ist. *`key`* kann jeder für einen Array-Index mögliche Wert sein.

Beispiel 1. array_key_exists()

```
$search_array = array("erstes" => 1, "zweites" => 4);
if (array_key_exists("erstes", $search_array)) {
    echo "Das Element 'erstes' ist in dem Array vorhanden";
}
```

Anmerkung: Der Name dieser Funktion ist in PHP 4.0.6 **key_exists()**.

Siehe auch isset().

array_keys (PHP 4 >= 4.0.0)

Liefert alle Schlüssel eines Arrays

array **array_keys** (array input [, mixed search_value])

array_keys() gibt die Schlüssel (numerisch und String) des Arrays *`input`* zurück.

Ist der optionale Parameter `search_value` angegeben, werden nur die Schlüssel für diesen Wert zurückgegeben. Andernfalls werden all Schlüssel von `input` zurückgegeben.

Beispiel 1. array_keys()

```
$array = array (0 => 100, "Farbe" => "rot");
print_r(array_keys ($array));

$array = array ("blau", "rot", "grün", "blau", "blau");
print_r(array_keys ($array, "blau"));

$array = array ("Farbe" => array("blau", "rot", "grün"), "Größe" => array("klein", "mit-
tel", "groß"));
print_r(array_keys ($array));
```

würde folgendes ausgeben:

```
Array
(
    [0] => 0
    [1] => Farbe
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => Farbe
    [1] => Größe
)
```

Anmerkung: Diese Funktion wurde in PHP 4 eingeführt, nachstehend finden Sie eine Implementierung für Benutzer von PHP 3.

Beispiel 2. Implementierung von array_keys() für Benutzer von PHP 3:

```
function array_keys ($arr, $term="") {
    $t = array();
    while (list($k,$v) = each($arr)) {
        if ($term && $v != $term) {
            continue;
        }
        $t[] = $k;
    }
    return $t;
```

```
}
```

Siehe auch array_values().

array_map (PHP 4 >= 4.0.6)

Wendet eine Callback-Funktion auf die Elemente von Arrays an

array **array_map** (mixed callback, array arr1 [, array arr2...])

array_map() gibt ein Array mit allen Elementen von *arr1* zurück, nachdem die Callback-Funktion auf jedes einzelne Element angewandt wurde. Die Anzahl Parameter, welche Callback-Funktion akzeptiert, sollte der Anzahl der an **array_map()** übergebenen Arrays entsprechen.

Beispiel 1. array_map()

```
function cube($n) {
    return $n*$n*$n;
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cube", $a);
print_r($b);
```

Dies speichert in $b:

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
```

Beispiel 2. array_map() - Verwendung mehrerer Arrays

```
function show_Spanish($n, $m) {
    return "Die Zahl $n heißt auf Spanisch  $m";
}

function map_Spanish($n, $m) {
    return array ($n => $m);
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("show_Spanish", $a, $b);
print_r($c);

$d = array_map("map_Spanish", $a , $b);
print_r($d);
```

Dies resultiert in:

```
// Ausgabe von $c
Array
(
    [0] => Die Zahl 1 heißt auf Spanisch  uno
    [1] => Die Zahl 2 heißt auf Spanisch  dos
    [2] => Die Zahl 3 heißt auf Spanisch  tres
    [3] => Die Zahl 4 heißt auf Spanisch  cuatro
    [4] => Die Zahl 5 heißt auf Spanisch  cinco
)

// Ausgabe von $d
Array
(
    [0] => Array
        (
            [1] => uno
        )

    [1] => Array
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )

    [4] => Array
```

```
        (
            [5] => cinco
        )

)
```

Bei Verwendung von zwei oder mehr Arrays sollten diese gewöhnlich die gleiche Länge besitzen, da die Callback-Funktion parallel auf die entsprechenden Elemente angewandt wird. Haben die Arrays unterschiedliche Längen, wird das kürzeste um leere Elemente erweitert.

Eine interessante Anwendung dieser Funktion ist die Konstruktion eines Arrays bestehend aus Arrays, was mit `NULL` als Name der Callback-Funktion leicht realisiert werden kann.

Beispiel 3. Erstellen eines Arrays mit Arrays

```
$a = array(1, 2, 3, 4, 5);
$b = array("eins", "zwei", "drei", "vier", "fünf");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
```

Würde folgendes ausgeben:

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => eins
            [2] => uno
        )

    [1] => Array
        (
            [0] => 2
            [1] => zwei
            [2] => dos
        )

    [2] => Array
        (
            [0] => 3
            [1] => drei
            [2] => tres
        )
```

```
    [3] => Array
        (
            [0] => 4
            [1] => vier
            [2] => cuatro
        )

    [4] => Array
        (
            [0] => 5
            [1] => fünf
            [2] => cinco
        )

)
```

Siehe auch array_filter() und array_reduce().

array_merge (PHP 4 >= 4.0.0)

Führt zwei oder mehr Arrays zusammen

array **array_merge** (array array1, array array2 [, array ...])

array_merge() fügt die Elemente von zwei oder mehr Arrays zusammen, indem die Werte des einen an das Ende des anderen angehängt werden. Das daraus resultierende Array wird zurückgegeben.

Haben die angegebenen Arrays die selben String-Schlüssel, dann wird der hintere Wert dieses Schlüssels den vorhergehenden überschreiben. Haben die Arrays den gleichen numerischen Schlüssel, dann wird der hintere Wert den vorhergehenden nicht überschreiben, sondern an den vorderen angehängt.

Beispiel 1. array_merge()

```
$array1 = array ("farbe" => "rot", 2, 4);
$array2 = array ("a", "b", "farbe" => "grün", "form" => "trapezoid", 4);
$result = array_merge ($array1, $array2);
```

Das `$result` Ergebnis ist:

```
Array
(
    [farbe] => grün
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
```

```
    [form] => trapezoid
    [4] => 4
)
```

Siehe auch array_merge_recursive().

array_merge_recursive (PHP 4)

Führt zwei oder mehr Arrays rekursiv zusammen

array **array_merge_recursive** (array array1, array array2 [, array ...])

array_merge_recursive() fügt die Elemente von zwei oder mehr Arrays zusammen, so dass die Werte des einen Arrays an die des vorigen angehängt werden. Das daraus resultierende Array wird zurückgegeben.

Haben die angegebenen Arrays die selben String-Schlüssel, dann wird der hintere Wert dieses Schlüssels den vorhergehenden überschreiben. Haben die Arrays den gleichen numerischen Schlüssel, dann wird der hintere Wert den vorhergehenden nicht überschreiben, sondern an den vorderen angehängt.

Beispiel 1. array_merge_recursive()

```
$ar1 = array ("farbe" => array ("favorit" => "rot"), 5);
$ar2 = array (10, "farbe" => array ("favorit" => "grün", "blau"));
$result = array_merge_recursive ($ar1, $ar2);
```

The `$result` will be:

```
Array
(
    [farbe] => Array
        (
            [favorit] => Array
                (
                    [0] => rot
                    [1] => grün
                )

            [0] => blau
        )

    [0] => 5
    [1] => 10
)
```

See also array_merge().

array_multisort (PHP 4 >= 4.0.0)

Sortiert mehrere oder multidimensionale Arrays

bool **array_multisort** (array ar1 [, mixed arg [, mixed ... [, array ...]]])

array_multisort() wird zum Sortieren von entweder mehreren Arrays auf einmal, oder eines multidimensionalen Arrays (entsprechend einer von mehreren Dimensionen) benutzt. Bei der Sortierung werden die Schlüsselassoziationen beibehalten.

Die angegebenen Arrays werden als Spalten einer Tabelle behandelt, welche nach Reihen sortiert werden - ähnlich der SQL Klausel ORDER BY. Das erste Array ist auch das erste in der Sortierreihenfolge. Die in diesem Array gleichen Zeilen (Werte) werden anhand des nächsten angegebenen Arrays sortiert, usw.

Die Struktur der Argumente ist etwas ungewöhnlich, aber flexibel. Das aller erste Argument muss ein Array sein. Die nachfolgenden Argumente können entweder ein Array oder eines der folgenden Sortierflags sein.

Flags für Sortierreihenfolge:

- SORT_ASC - sortiere in aufsteigender Reihenfolge
- SORT_DESC - sortiere in absteigender Reihenfolge

Flags für Sortiertypen:

- SORT_REGULAR - vergleiche Felder normal
- SORT_NUMERIC - vergleiche Felder numerisch
- SORT_STRING - vergleiche Felder als Strings

Es kann nur ein Sortierflag des selben Typs nach jedem Array spezifiziert werden. Sortierflags nach einem Array Argument gelten nur für dieses Array, und werden vor jedem neuen Array Argument zu den Defaultflags SORT_ASC und SORT_REGULAR zurückgesetzt.

Liefert bei Erfolg `TRUE`, und im Fehlerfall `FALSE`.

Beispiel 1. Sortieren mehrerer Arrays

```
$ar1 = array ("10", 100, 100, "a");
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

In diesem Beispiel enthält das erste Array nach dem Sortieren 10, "a", 100, 100. Das zweite Array wird 1, 1, "2", 3 enthalten. Die Einträge des zweiten Arrays, welche den identischen Einträgen des ersten Arrays entsprechen (100 und 100) wurden ebenfalls sortiert.

Beispiel 2. Sortieren eines mehrdimensionalen Arrays

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

In diesem Beispiel enthält das erste Array nach dem Sortieren 10, 100, 100, "a" (wurde als Strings in aufsteigender Reihenfolge sortiert), und das zweite enthält 1, 3, "2", 1 (numerisch und absteigender Reihenfolge sortiert).

array_pad (PHP 4 >= 4.0.0)

Vergrößert ein Array auf die spezifizierte Länge mit einem Wert

array **array_pad** (array input, int pad_size, mixed pad_value)

array_pad() liefert eine Kopie von *input*, welche auf die von *pad_size* spezifizierte Größe mit dem Wert *pad_value* erweitert wurde. Ist der Parameter *pad_size* positiv, wird das Array rechts erweitert, ist er negativ, dann erfolgt die Erweiterung links. Ist der absolute Wert von *pad_size* kleiner oder gleich der Länge von *input*, erfolgt keine Erweiterung.

Beispiel 1. array_pad()

```
$input = array (12, 10, 9);

$result = array_pad ($input, 5, 0);
// Ergebnis ist array (12, 10, 9, 0, 0)

$result = array_pad ($input, -7, -1);
// Ergebnis ist array (-1, -1, -1, -1, 12, 10, 9)

$result = array_pad ($input, 2, "noop");
// nicht erweitert
```

array_pop (PHP 4 >= 4.0.0)

Liefert das letzte Element eines Arrays

mixed **array_pop** (array array)

array_pop() liefert den letzten Wert von *array*, und verkürzt *array* um ein Element. Ist *array* leer (oder kein Array), wird `NULL` zurückgegeben.

Beispiel 1. array_pop()

```
$stack = array ("Orange", "Banane", "Apfel", "Himbeere");
$fruit = array_pop ($stack);
```

Danach hat `$stack` nur 3 Elemente:

```
Array
(
    [0] => Orange
    [1] => Banane
    [2] => Apfel
)
```

und `rasberry` wird `$fruit` zugewiesen.

Warnung

Diese Function kann `FALSE` zurückliefern, kann aber auch einen Wert zurückgeben, der von einem einfachen if-Statement als `FALSE` ausgewertet wird. Benutzen Sie deshalb bitte den === Operator um den Rückgabewert dieser Funktion zu überprüfen.

Siehe auch array_push(), array_shift() und array_unshift().

array_push (PHP 4 >= 4.0.0)

Fügt ein oder mehr Elemente an das Ende eines Arrays

int **array_push** (array array, mixed var [, mixed ...])

array_push() behandelt *array* als Stapel (Stack), und fügt die übergebenen Variablen an das Ende von *array*. Die Länge von *array* wird dabei um die Anzahl der angefügten Variablen erhöht. Dies hat den selben Effekt wie:

```
$array[] = $var;
```

für jede Variable *var* wiederholt.

Liefert die neue Anzahl Elemente des Arrays.

Beispiel 1. array_push()

```
$stack = array ("Orange", "Banane");
array_push ($stack, "Apfel", "Himbeere");
```

Danach enthält `$stack` die folgenden Elemente:

```
Array
(
    [0] => Orange
    [1] => Banane
    [2] => Apfel
    [3] => Himbeere
)
```

Siehe auch array_pop(), array_shift() und array_unshift().

array_rand (PHP 4 >= 4.0.0)

Liefert einen oder mehrere zufällige Einträge eines Arrays

mixed **array_rand** (array input [, int num_req])

array_rand() ist ziemlich nützlich, wenn Sie einen oder mehrere zufällige Einträge eines Arrays auswählen möchten. Die Funktion übernimmt das Array *input*, und ein optionales Argument *num_req*, welches die gewünschte Anzahl Einträge spezifiziert. Ist *num_req* nicht angegeben, wird ein Defaultwert von 1 angenommen.

Wenn Sie nur einen Eintrag auswählen, liefert **array_rand()** den Schlüssel eines zufälligen Eintrages. Andernfalls wird ein Array mit den Schlüsseln der zufälligen Einträge zurückgegeben. Dies hat den Zweck, dass Sie zufällige Schlüssel und auch Werte aus dem Array auswählen können.

Vergessen Sie bitte nicht, mittels srand() den Startwert für den Zufallsgenerator festzulegen.

Beispiel 1. array_rand()

```
srand ((float) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$rand_keys = array_rand ($input, 2);
```

```
print $input[$rand_keys[0]]."\n";
print $input[$rand_keys[1]]."\n";
```

array_reverse (PHP 4 >= 4.0.0)

Liefert ein Array mit umgekehrter Reihenfolge der Elemente

array **array_reverse** (array array [, bool preserve_keys])

array_reverse() liefert ein Array mit der umgekehrten Reihenfolge der Elemente von *array*. Ist der optionale Parameter *preserve_keys* TRUE, bleibt die Reihenfolge der Schlüssel aufrecht.

Beispiel 1. array_reverse()

```
$input  = array ("php", 4.0, array ("grün", "rot"));
$result = array_reverse ($input);
$result_keyed = array_reverse ($input, TRUE);
```

Danach haben $result und $result_keyed die gleichen Elemente, aber Beachten Sie den Unterschied zwischen den Schlüsseln. Die Ausgabe von $result und $result_keyed wird wie folgt aussehen:

```
Array
(
    [0] => Array
        (
            [0] => grün
            [1] => rot
        )

    [1] => 4
    [2] => php
)
Array
(
    [2] => Array
        (
            [0] => grün
            [1] => rot
        )

    [1] => 4
    [0] => php
)
```

Anmerkung: Der zweite Parameter wurde in PHP 4.0.3 hinzugefügt.

array_reduce (PHP 4 >= 4.0.5)

Iterative Reduktion eines Arrays zu einem Wert mittels einer Callback Funktion

mixed **array_reduce** (array input, mixed callback [, int initial])

array_reduce() wendet die *callback* Funktion iterativ bei den Elementen des Arrays *input* so an, dass das Array auf einen einzigen Wert reduziert wird. Ist der optionale Parameter *intial* angegeben, wird er am Beginn des Prozesses, benutzt oder als Resultat verwendet, sollte das Array leer sein.

Beispiel 1. array_reduce()

```
function rsum($v, $w) {
    $v += $w;
    return $v;
}

function rmul($v, $w) {
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
```

Hier enthält `$b` `15`, `$c` `1200` (= 1*2*3*4*5*10), und `$d` enthält `1`.

Siehe auch array_filter() und array_map().

array_shift (PHP 4 >= 4.0.0)

Liefert ein Element vom Beginn eines Arrays

mixed **array_shift** (array array)

array_shift() liefert den ersten Wert von *array*, verschiebt die anderen Werte hinunter, und verkürzt *array* um ein Element. Ist *array* leer (oder kein Array), wird `NULL` zurückgegeben.

Beispiel 1. array_shift()

```
$stack = array ("Orange", "Banane", "Apfel", "Himbeere");
$fruit = array_shift ($stack);
```

Danach blieben in `$stack` 3 Elemente übrig:

```
Array
(
    [0] => Banane
    [1] => Apfel
    [2] => Himbeere
)
```

und `orange` wird `$fruit` zugeordnet.

Siehe auch array_unshift(), array_push() und array_pop().

array_slice (PHP 4 >= 4.0.0)

Extrahiert einen Ausschnitt eines Arrays

array **array_slice** (array array, int offset [, int length])

array_slice() liefert die mittels *`offset`* und *`length`* spezifizierte Sequenz von Elementen des Arrays *`array`*

Ist *`offset`* positiv, beginnt die Sequenz bei diesem Offset in dem *`array`*. Ist *`offset`* negativ, beginnt die Sequenz *`offset`* Elemente vor dem Ende von *`array`*.

Ist *`length`* positiv, enthält die Sequenz genauso viele Elemente. Ist *`length`* negativ, enthält die Sequenz alle Elemente des Arrays von *`offset`* bis *`length`* Elemente vor dem Ende des Arrays. Ist *`length`* nicht angegeben, enthält die Sequenz alle Elemente von *`offset`* bis zum Ende von *`array`*.

Beachten Sie, dass **array_slice()** Schlüssel ignoriert, und Offsets sowie Längen anhand der aktuellen Positionen berechnet.

Beispiel 1. array_slice()

```
$input = array ("a", "b", "c", "d", "e");

$output = array_slice ($input, 2);      // liefert "c", "d", und "e"
$output = array_slice ($input, 2, -1);  // liefert "c", "d"
$output = array_slice ($input, -2, 1);  // liefert "d"
$output = array_slice ($input, 0, 3);   // liefert "a", "b", und "c"
```

Siehe auch array_splice().

array_splice (PHP 4 >= 4.0.0)

Entfernt einen Teil eines Arrays und ersetzt ihn durch etwas anderes

array **array_splice** (array input, int offset [, int length [, array replacement]])

array_splice() entfernt die durch *offset* und *length* angegebenen Elemente des Arrays *input*, und ersetzt diese durch die Elemente des Arrays *replacement*, wenn angegeben und gibt ein Array mit den entfernten Elemente zurück.

Ist *offset* positiv, beginnt der zu entfernende Bereich bei diesem Offset vom Anfang des Arrays *input*. Ist *offset* negativ, beginnt der zu entfernende Bereich *offset* Elemente vor dem Ende des Arrays *input*.

Ist *length* nicht angegeben, wird alles von *offset* bis zum Ende des Arrays entfernt. Ist *length* positiv, wird die angegebene Anzahl Elemente entfernt. Ist *length* negativ, dann wird der Bereich von *length* Elementen vor dem Ende, bis zum Ende des Arrays entfernt. Tipp: Um alles von *offset* bis zum Ende des Arrays zu entfernen wenn *replacement* ebenfalls angegeben ist, verwenden Sie `count($input)` als *length*.

Ist das Array *replacement* angegeben, werden die entfernten Elemente durch die Elemente dieses Arrays ersetzt. Sind *offset* und *length* so angegeben dass nichts entfernt wird, werden die Elemente von *replacement* an der von *offset* spezifizierten Stelle eingefügt. Tipp: Soll die Ersetzung durch nur ein Element erfolgen ist es nicht nötig ein Array zu anzugeben es sei denn, dieses Element ist selbst ein Array.

Hier einige Entsprechungen:

```
array_push ($input, $x, $y)     array_splice ($input, count ($input), 0,
                                             array ($x, $y))
array_pop ($input)              array_splice ($input, -1)
array_shift ($input)            array_splice ($input, 0, 1)
array_unshift ($input, $x, $y)  array_splice ($input, 0, 0, array ($x, $y))
$a[$x] = $y                     array_splice ($input, $x, 1, $y)
```

Gibt das Array mit den entfernten Elementen zurück.

Beispiel 1. array_splice()

```
$input = array ("rot", "grün", "blau", "gelb");
array_splice ($input, 2);
// $input ist nun array ("rot", "grün")

$input = array ("rot", "grün", "blau", "gelb");
```

```
array_splice ($input, 1, -1);
// $input ist nun array ("rot", "gelb")

$input = array ("rot", "grün", "blau", "gelb");
array_splice ($input, 1, count($input), "orange");
// $input ist nun array ("rot", "orange")

$input = array ("rot", "grün", "blau", "gelb");
array_splice ($input, -1, 1, array("schwarz", "braun"));
// $input ist nun array ("rot", "grün", "blau", "schwarz", "braun")
```

Siehe auch array_slice().

array_sum (PHP 4 >= 4.0.4)

Liefert die Summe der Werte in einem Array

mixed **array_sum** (array array)

array_sum() liefert die Summe der Werte eines Arrays als Integer oder Float.

Beispiel 1. array_sum()

```
$a = array(2, 4, 6, 8);
echo "sum(a) = ".array_sum($a)."\n";

$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "sum(b) = ".array_sum($b)."\n";
```

Obiges Programm wird folgendes ausgeben:

```
sum(a) = 20
sum(b) = 6.9
```

Anmerkung: PHP Versionen vor 4.0.6 haben das übergebene Array selbst modifiziert, und Strings in Zahlen konvertiert (welche abhängig von deren Wert meist zu null konvertiert wurden).

array_unique (PHP 4)

Entfernt doppelte Werte aus einem Array

array **array_unique** (array array)

array_unique() nimmt *array* und gibt ein neues Array zurück, aus dem alle doppelten Einträge entfernt wurden.

Beachten Sie, dass Schlüssel bewahrt bleiben. Erst behandelt **array_unique()** die Werte als Strings und sortiert sie, danach wird der erste gefundene Schlüssel behalten, und alle folgenden Schlüssel ignoriert. Das heißt nicht, dass der Schlüssel des ersten zugehörigen Wertes aus dem unsortierten *array* behalten wird.

Anmerkung: Zwei Elemente werden nur dann als gleich angesehen, wenn `(string) $elem1 === (string) $elem2`. In Worten: Wenn die String-Repräsentation die gleiche ist.

Das erste Element wird verwendet.

Warnung

Dies war in PHP 4.0.4 fehlerhaft!

Beispiel 1. array_unique() example

```
$input = array ("a" => "grün", "rot", "b" => "grün", "blau", "rot");
$result = array_unique ($input);
print_r($result);
```

Dieses Programm wird folgendes ausgeben:

```
Array
(
    [b] => grün
    [1] => blau
    [2] => rot
)
```

Beispiel 2. array_unique() und Typen

```
$input = array (4,"4","3",4,3,"3");
$result = array_unique ($input);
var_dump($result);
```

Dieses Programm wird folgendes ausgeben (PHP 4.0.6):

```
array(2) {
  [3]=>
  int(4)
  [4]=>
  int(3)
}
```

array_unshift (PHP 4 >= 4.0.0)

Fügt ein oder mehr Elemente am Anfang eines Arrays ein

int **array_unshift** (array array, mixed var [, mixed ...])

array_unshift() fügt die übergebenen Elemente am Anfang von *array* ein. Beachten Sie, dass die Liste von Elementen als Ganzes eingefügt wird, sodass die eingefügten Elemente die selbe Reihenfolge haben.

Diese Funktion gibt die neue Anzahl der in *array* enthaltenen Elemente zurück.

Beispiel 1. array_unshift()

```
$queue = array ("Orange", "Banane");
array_unshift ($queue, "Apfel", "Himbeere");
```

Als Ergebnis enthält `$queue` die folgenden Elemente:

```
Array
(
    [0] => Apfel
    [1] => Himbeere
    [2] => Orange
    [3] => Banane
)
```

Siehe auch array_shift(), array_push() und array_pop().

array_values (PHP 4 >= 4.0.0)

Liefert alle Werte eines Arrays

array **array_values** (array input)

array_values() liefert alle Werte des Arrays *input*.

Beispiel 1. array_values()

```
$array = array ("größe" => "XL", "farbe" => "gold");
print_r(array_values ($array));
```

Dies wird folgendes ausgeben:

```
Array
(
    [0] => XL
    [1] => gold
)
```

Anmerkung: Diese Funktion wurde in PHP 4 hinzugefügt, nachstehend findet sich eine Implementation für Benutzer von PHP 3.

Beispiel 2. Implementation von array_values() für PHP 3 Benutzer

```
function array_values ($arr) {
    $t = array();
    while (list($k, $v) = each ($arr)) {
        $t[] = $v;
    }
    return $t;
}
```

Siehe auch array_keys().

array_walk (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Wendet eine Benutzerfunktion an jedem Element eines Arrays an

int **array_walk** (array array, string func [, mixed userdata])

Wendet die mittels *func* angegebene Benutzerfunktion für jedes Element von *arr* an. *func* wird der Wert als erster, und der Schlüssel als zweiter Parameter übergeben. Sofern angegeben, wird *userdata*, als dritter Parameter an die Funktion *func* übergeben. *func* muss eine vom Benutzer definierte Funktion sein, und nicht eine native PHP Funktion. Deshalb können Sie **array_walk()** nicht direkt mit **str2lower()** benutzen, sondern müssen erst eine eigene Funktion damit erstellen, und diese Funktion als Argument übergeben.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

Sollte *func* mehr als zwei oder drei Argumente verlangen, wird - abhängig von *userdata* - bei jedem Aufruf von *func* seitens **array_walk()** eine Warnmeldung generiert. Diese Warnmeldungen können unterdrückt werden, indem man dem Funktionsaufruf von **array_walk()** ein '@' Zeichen voranstellt, oder mittels der Verwendung von error_reporting().

Anmerkung: Wenn *func* mit den Werten des Arrays direkt arbeiten soll, geben Sie an, dass der erste Parameter von *func* als Referenz übergeben werden muss. Dann werden alle Änderungen an diesen Elementen direkt in dem selbst durchgeführt.

Die Modifizierung des Arrays aus *func* heraus könnte ein unvorhersehbares Verhalten hervorrufen.

Anmerkung: Die Übergabe des Schlüssels und von *userdata* an *func* wurde mit PHP 4.0 eingeführt.

In PHP 4 muss reset() aufgerufen werden, da **array_walk()** den internen Zeiger nicht selbsttätig wieder auf das erste Element zurücksetzt.

Beispiel 1. array_walk()

```
$fruits = array ("d"=>"Zitrone", "a"=>"Orange", "b"=>"Banane", "c"=>"Apfel");

function test_alter (&$item1, $key, $prefix) {
    $item1 = "$prefix: $item1";
}

function test_print ($item2, $key) {
    echo "$key. $item2<br>\n";
}
echo "Vorher ...:\n";
array_walk ($fruits, 'test_print');
reset ($fruits);
array_walk ($fruits, 'test_alter', 'Frucht');
```

```
echo "... und nachher:\n";
reset ($fruits);
array_walk ($fruits, 'test_print');
```

Das obige Programm wird folgendes ausgeben:

```
Vorher ...:
d. Zitrone
a. Orange
b. Banane
c. Apfel
... und nachher:
d. Frucht: Zitrone
a. Frucht: Orange
b. Frucht: Banane
c. Frucht: Apfel
```

Siehe auch each() und list().

arsort (PHP 3, PHP 4 >= 4.0.0)

Sortiert ein Array in umgekehrter Reihenfolge und behält die Verbindung zum Index

void **arsort** (array array [, int sort_flags])

Diese Funktion sortiert ein Array so, dass der Zusammenhang zwischen den Indizes und den entsprechenden Arrayelementen erhalten bleibt. Dies wird hauptsächlich bei assoziativen Arrays verwendet, bei denen die aktuelle Reihenfolge der Elemente bedeutend ist.

Beispiel 1. arsort()

```
$fruits = array ("d"=>"Zitrone", "a"=>"Orange", "b"=>"Banane", "c"=>"Apfel");
arsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Dieses Beispiel würde folgendes ausgeben:

```
d = Zitrone
a = Orange
b = Banane
c = Apfel
```

Die Früchte wurden in umgekehrter alphabetischer Reihenfolge sortiert, und die Zuordnung zwischen Index und Element blieb beibehalten.

Sie können das Verhalten der Sortierung mittels dem optionalen Parameter *sort_flags* beeinflussen, für Details siehe sort().

Siehe auch asort(), rsort(), ksort() und sort().

asort (PHP 3, PHP 4 >= 4.0.0)

Sortiert ein Array und behält die Indexverbindungen

void **asort** (array array [, int sort_flags])

Diese Funktion sortiert ein Array so, dass der Zusammenhang zwischen den Indizes und den entsprechenden Arrayelementen erhalten bleibt. Dies wird hauptsächlich bei assoziativen Arrays verwendet, bei denen die aktuelle Reihenfolge der Elemente bedeutend ist.

Beispiel 1. asort()

```
$fruits = array ("d"=>"Zitrone", "a"=>"Orange", "b"=>"Banane", "c"=>"Apfel");
asort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Dieses Beispiel würde folgendes ausgeben:

```
c = Apfel
b = Banane
a = Orange
d = Zitrone
```

Die Früchte wurden alphabetisch sortiert, und die Zuordnung zwischen Index und Element blieb beibehalten

Sie können das Verhalten der Sortierung mittels dem optionalen Parameter *sort_flags* beeinflussen, für Details siehe sort().

Siehe auch arsort(), rsort(), ksort() und sort().

compact (PHP 4 >= 4.0.0)

Erstellt ein Array mit Variablen und deren Werten

array **compact** (mixed varname [, mixed ...])

compact() übernimmt eine variable Anzahl von Parametern. Jeder Parameter kann entweder ein String mit einem Variablennamen, oder ein Array mit Variablennamen sein. Dieses Array kann auch andere Arrays mit Variablennamen enthalten; **compact()** behandelt sie rekursiv.

Für alle diese sucht **compact()** nach einer Variablen in der aktuellen Symboltabelle, und fügt diese dem zurückzugebenden Array hinzu, wobei der Variablenname als Schlüssel, und der Inhalt der Variablen als Wert gespeichert wird. Kurz, diese Funktion tut das Gegenteil von extract(). Sie gibt ein Array mit allen Variablen zurück.

Strings, welche nicht gesetzt sind, werden einfach ignoriert.

Beispiel 1. compact()

```
$city = "San Francisco";
$state = "CA";
$event = "SIGGRAPH";

$location_vars = array ("city", "state");

$result = compact ("event", "nothing_here", $location_vars);
```

Danach wird `$result`:

```
Array
(
    [event] => SIGGRAPH
    [city] => San Francisco
    [state] => CA
)
```

Siehe auch extract().

count (PHP 3, PHP 4 >= 4.0.0)

Zählt die Elemente einer Variable

int **count** (mixed var)

Liefert die Anzahl von Elementen in *`var`*, welche typischerweise ein Array ist (da alles andere ein Element enthält).

Ist `var` kein Array, wird `1` zurückgegeben (Ausnahme: `count(NULL)` ist gleich `0`).

Warnung

count() kann 0 zurückgeben wenn die Variable nicht gesetzt ist, aber sie könnte ebenfalls 0 zurückgeben, wenn eine Variable als leeres Array initialisiert wurde. Verwenden Sie isset() um zu testen, ob eine Variable gesetzt ist.

In der Sektion Arrays finden Sie eine detaillierte Erklärung, wie Arrays in PHP implementiert sind und wie sie benutzt werden.

Beispiel 1. count()

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count ($a);
//$result == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result = count ($b);
// $result == 3;
```

Anmerkung: Die Funktion sizeof() ist ein Alias für **count()**.

Siehe auch is_array(), isset() und strlen().

current (PHP 3, PHP 4 >= 4.0.0)

Liefert das aktuelle Element eines Arrays

mixed **current** (array array)

Jedes Array hat einen internen Zeiger auf sein "aktuelles" Element, welcher auf das erste in das Array eingefügte Element initialisiert wird.

Die Funktion **current()** liefert das Array Element, auf das gerade vom internen Zeiger gezeigt wird. Sie bewegt den Zeiger in keinster Weise. Zeigt der interne Zeiger hinter das Ende der Elementenliste, gibt **current()** `FALSE` zurück.

Warnung

Enthält das Array leere Elemente (0 oder "", den leeren String), gibt diese Funktion auch für diese Elemente FALSE zurück. Das macht es unmöglich, unter Verwendung von **current()** herauszufinden, ob das wirklich am Ende der Liste sind. Um ein Array, das leere Elemente enthalten könnte richtig durchzugehen, verwenden Sie die Funktion each().

Siehe auch end(), next(), prev() und reset().

each (PHP 3, PHP 4 >= 4.0.0)

Liefert das aktuelle Paar (Schlüssel und Wert) eines Arrays und rückt den Array Cursor vor.

array **each** (array array)

Liefert das aktuelle Schlüssel- Wertepaar des Arrays *array*, und erhöht den Array Cursor. Dieses Paar wird als Array mit 4 Elementen zurückgegeben, und zwar mit den Schlüsseln *0*, *1*, *key*, und *value*. Die Elemente *0* und *key* enthalten den Namen des Schlüssels, *1* und *value* den Wert.

Zeigt der interne Zeiger hinter das Ende der Array-Inhalte, gibt **each()** FALSE zurück.

Beispiel 1. each() Beispiele

```
$foo = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each ($foo);
```

$bar enthält nun die folgenden Schlüssel/Wert Paare:

- 0 => 0
- 1 => 'bob'
- key => 0
- value => 'bob'

```
$foo = array ("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each ($foo);
```

$bar enthält nun die folgenden Schlüssel/Wert Paare:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'
- value => 'Bob'

each() wird typischerweise in Verbindung mit list() verwendet, um ein Array durchzugehen; zum Beispiel `$HTTP_POST_VARS`:

Beispiel 2. Durchgehen aller `$HTTP_POST_VARS` mit each()

```
echo "Werte, die mit der POST Methode übertragen wurden:<br>";
reset ($HTTP_POST_VARS);
while (list ($key, $val) = each ($HTTP_POST_VARS)) {
    echo "$key => $val<br>";
}
```

Nachdem **each()** ausgeführt wurde, bleibt der Array Cursor beim nächsten Element des Arrays, oder wenn am Ende des Arrays angelangt, bei dem letzten Element. Sie müssen reset() verwenden, wenn Sie das Array erneut mittels each durchgehen wollen.

Siehe auch key(), list(), current(), reset(), next() und prev().

end (PHP 3, PHP 4 >= 4.0.0)

Positioniert den internen Zeiger eines Arrays auf dessen letztes Element

mixed **end** (array array)

end() rückt den internen Zeiger von *`array`* bis zum letzten Element vor, und gibt dieses Element zurück.

Siehe auch current(), each(), next() und reset().

extract (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Importiert Variablen in die aktuelle Symboltabelle eines Arrays

int **extract** (array var_array [, int extract_type [, string prefix]])

Diese Funktion wird verwendet, um Variablen eines Arrays in die aktuelle Symboltabelle zu importieren. Es behandelt die Schlüssel des assoziativen Arrays *`var_array`* als Variablennamen, und die Werte als Variablenwerte. Für jedes Paar Schlüssel/Wert wird - abhängig von den Parametern *`extract_type`* und *`prefix`* - eine Variable in der aktuellen Symboltabelle eingetragen.

Anmerkung: Seit Version 4.0.5 gibt diese Funktion die Anzahl der extrahierten Variablen zurück.

Anmerkung: EXTR_IF_EXISTS und EXTR_PREFIX_IF_EXISTS wurde in Version 4.2.0. eingeführt.

extract() prüft jeden Schlüssel, ob dieser einen gültigen Variablennamen ergibt, und auch auf eventuelle Kollisionen mit existierenden Variablen in der Symboltabelle. Die Art, wie ungültige/numerische Schlüssel und Kollisionen behandelt werden, wird mit *extract_type* spezifiziert. Es kann einen der folgenden Werte annehmen:

EXTR_OVERWRITE

Im Falle einer Kollision wird die existierende Variable überschrieben.

EXTR_SKIP

Im Falle einer Kollision wird die existierende Variable nicht überschrieben.

EXTR_PREFIX_SAME

Im Falle einer Kollision wird dem Variablennamen *prefix* vorangestellt.

EXTR_PREFIX_ALL

Allen Variablennamen wird *prefix* vorangestellt. Seit PHP 4.0.5 gilt dies auch für numerische Schlüssel.

EXTR_PREFIX_INVALID

Nur ungültigen/numerischen Variablennamen wird *prefix* vorangestellt. Dieser Flag wurde mit PHP 4.0.5 eingeführt.

EXTR_IF_EXISTS

Überschreibt die Variable nur, wenn sie bereits in der aktuellen Symboltabelle existiert, sonst geschieht nichts. Dies ist dann hilfreich, wenn Sie eine Liste mit gültigen Variablen definieren, und dann z.B. nur jene Variablen extrahieren, welche Sie aus $_REQUEST definiert haben. Dieser Flag wurde in PHP 4.2.0 eingeführt.

EXTR_PREFIX_IF_EXISTS

Erstellt nur Variablennamen mit Präfix, wenn die Version ohne Präfix der selben Variable in der aktuellen Symboltabelle existiert. Dieser Flag wurde in PHP 4.2.0 eingeführt.

Ist *extract_type* nicht spezifiziert, so wird EXTR_OVERWRITE angenommen.

Beachten Sie, dass *prefix* nur dann nötig ist, wenn *extract_type* EXTR_PREFIX_SAME, EXTR_PREFIX_ALL, EXTR_PREFIX_INVALID, oder EXTR_PREFIX_IF_EXISTS ist. Ist das Resultat trotz Präfix kein gültiger Variablenname, wird es nicht in die Symboltabelle importiert.

extract() gibt die Anzahl erfolgreich in die Symboltabelle importierter Variablen zurück.

Eine mögliche Verwendung für extract ist der Import der Variablen in einem von wddx_deserialize() gelieferten assoziativen Array in die Symboltabelle.

Beispiel 1. extract()

```
<?php

/* Annahme: $var_array ist ein von wddx_deserialize
```

```
   zurückgegebenes Array */

$groesse = "groß";
$var_array = array ("farbe"   => "blau",
                    "groesse" => "mittel",
                    "form"    => "Kugel");
extract ($var_array, EXTR_PREFIX_SAME, "wddx");

print "$farbe, $groesse, $form, $wddx_groesse\n";

?>
```

Das obige Beispiel wird folgendes ausgeben:

```
blau, groß, Kugel, mittel
```

`$size` wurde nicht überschrieben, da wir EXTR_PREFIX_SAME spezifiziert haben, was im Anlegen von `$wddx_size` endete. Wäre EXTR_SKIP spezifiziert worden, dann wäre $wddx_size nicht angelegt worden. EXTR_OVERWRITE hätte `$size` mit dem Wert "medium" überschrieben, und EXTR_PREFIX_ALL hätte die neuen Variablen `$wddx_color`, `$wddx_size`, und `$wddx_shape` angelegt.

Sie müssen assoziative Arrays benutzen, numerisch indizierte Arrays liefern keine Resultate.

Siehe auch compact().

in_array (PHP 4 >= 4.0.0)

Prüft, ob ein Wert in einem Array existiert

bool **in_array** (mixed needle, array haystack [, bool strict])

Diese Funktion sucht in *`haystack`* nach *`needle`* und gibt bei Erfolg `TRUE` zurück, andernfalls `FALSE`.

Ist der dritte Parameter *`strict`* auf `TRUE` gesetzt, prüft **in_array()** auch die Typen von *`needle`* in *`haystack`*.

Anmerkung: Ist *`needle`* ein String, erfolgt der Vergleich unter Berücksichtigung der Groß- und Kleinschreibung.

Anmerkung: In den PHP Versionen vor 4.2.0 durfte *`needle`* kein Array sein

Beispiel 1. in_array()

```
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os)) {
    print "Got Irix";
}
if (in_array ("mac", $os)) {
    print "Got mac";
}
```

Die zweite Bedingung schlägt fehl, da **in_array()** die Groß-/Kleinschreibung berücksichtigt, daher wird das obige Programm folgendes ausgeben:

```
Got Irix
```

Beispiel 2. in_array() mit strict

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, TRUE))
    echo "'12.4' gefunden mit strict check\n";
if (in_array(1.13, $a, TRUE))
    echo "1.13 gefunden mit strict check\n";
?>
```

Dies wird folgendes anzeigen:

```
1.13 gefunden mit strict check
```

Beispiel 3. in_array() mit einem Array als needle

```
<?php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if (in_array(array ('p', 'h'), $a))
    echo "'ph' wurde gefunden\n";
if (in_array(array ('f', 'i'), $a))
    echo "'fi' wurde nicht gefunden\n";
```

```
if (in_array('o', $a))
    echo "'o' wurde gefunden\n";
?>

// Dies wird folgendes anzeigen:

'ph' wurde gefunden
'o' wurde gefunden
```

Siehe auch array_search().

array_search (PHP 4 >= 4.0.5)

Durchsucht ein Array nach einem Wert liefert bei Erfolg den Schlüssel

mixed **array_search** (mixed needle, array haystack [, bool strict])

Diese Funktion durchsucht *haystack* nach *needle* und gibt bei Erfolg den Schlüssel zurück, andernfalls FALSE.

Ist der optionale dritte Parameter *strict* auf TRUE gesetzt, prüft in_array() auch die Typen von *needle* in *haystack*.

> **Warnung**
>
> Diese Function kann FALSE zurückliefern, kann aber auch einen Wert zurückgeben, der von einem einfachen if-Statement als FALSE ausgewertet wird. Benutzen Sie deshalb bitte den === Operator um den Rückgabewert dieser Funktion zu überprüfen.

Siehe auch in_array().

key (PHP 3, PHP 4 >= 4.0.0)

Liefert einen Schlüssel eines assoziativen Arrays

mixed **key** (array array)

key() liefert das Index Element der aktuellen Position im Array.

Siehe auch current() und next().

krsort (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sortiert ein Array nach Schlüsseln in umgekehrter Reihenfolge

int **krsort** (array array [, int sort_flags])

Sortiert ein Array nach Schlüsseln in umgekehrter Reihenfolge, wobei die Zuordnungen zwischen Schlüssel und Wert erhalten bleiben. Dies ist hauptsächlich bei assoziativen Arrays hilfreich.

Beispiel 1. krsort()

```
$fruits = array ("d"=>"Zitrone", "a"=>"Orange", "b"=>"Banane", "c"=>"Apfel");
krsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Dieses Beispiel würde folgendes ausgeben:

```
d = Zitrone
c = Apfel
b = Banane
a = Orange
```

Sie können das Verhalten der Sortierung mittels dem optionalen Parameter *sort_flags* beeinflussen, für Details siehe sort().

Siehe auch asort(), arsort(), ksort(), sort(), natsort() und rsort().

ksort (PHP 3, PHP 4 >= 4.0.0)

Sortiert ein Array nach Schlüsseln

int **ksort** (array array [, int sort_flags])

Sortiert ein Array nach Schlüsseln, wobei die Zuordnungen zwischen Schlüssel und Wert erhalten bleiben. Dies ist hauptsächlich bei assoziativen Arrays hilfreich.

Beispiel 1. ksort()

```
$fruits = array ("d"=>"Zitrone", "a"=>"Orange", "b"=>"Banane", "c"=>"Apfel");
ksort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Dieses Beispiel würde folgendes ausgeben:

```
a = Orange
b = Banane
c = Apfel
d = Zitrone
```

Sie können das Verhalten der Sortierung mittels dem optionalen Parameter *sort_flags* beeinflussen, für Details siehe sort().

Siehe auch asort(), arsort(), krsort(), uksort(), sort(), natsort() und rsort().

Anmerkung: Der zweite Parameter wurde mit PHP 4 eingeführt.

list (unknown)

Weist Variablen zu, als wären sie ein Array

void **list** (mixed ...)

Wie array() ist auch dies keine wirkliche Funktion, sondern ein Sprachkonstrukt. **list()** wird verwendet, um eine Liste von Variablen innerhalb einer Operation zuzuweisen.

Beispiel 1. list()

```
<table>
 <tr>
  <th>Mitarbeitername</th>
  <th>Gehalt</th>
 </tr>

<?php

$result = mysql_query ("SELECT id, name, salary FROM employees",$conn);
while (list ($id, $name, $salary) = mysql_fetch_row ($result)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php?id=$id\">$name</a></td>\n".
           "  <td>$salary</td>\n".
           " </tr>\n");
}

?>

</table>
```

Siehe auch each() und array().

natsort (PHP 4 >= 4.0.0)

Sortiert ein Array in "natürlicher Reihenfolge"

void **natsort** (array array)

Diese Funktion implementiert einen Sortieralgorithmus, welcher alphanumerische Zeichenketten reiht, wie es auch ein Mensch tun würde. Hier ein Beispiel für den Unterschied zwischen diesem und dem normalen Sortieralgorithmus für Strings eines Computers (verwendet in sort()):

Beispiel 1. natsort()

```
$array1 = $array2 = array ("img12.png", "img10.png", "img2.png", "img1.png");
sort ($array1);
echo "Standardsortierung\n";
print_r ($array1);

natsort ($array2);
echo "\nSortieren mit natürlicher Reihenfolge\n";
print_r ($array2);
```

Der obige Code wird folgendes anzeigen:

```
Standardsortierung
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Sortieren mit natürlicher Reihenfolge
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Detailliertere Informationen finden Sie auf Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) Seite.

Siehe auch natcasesort(), strnatcmp() und strnatcasecmp().

natcasesort (PHP 4 >= 4.0.0)

Sortiert ein Array in "natürlicher Reihenfolge", Groß/Kleinschreibung wird ignoriert

void **natcasesort** (array array)

Diese Funktion implementiert einen Sortieralgorithmus, welcher alphanumerische Zeichenketten reiht, wie es auch ein Mensch tun würde. Das wird als "natürliche Reihenfolge" bezeichnet.

natcasesort() arbeitet wie natsort(), nur dass es die Groß/Kleinschreibung ignoriert. Ein Beispiel für den Unterschied zwischen diesem und dem normalen Sortieralgorithmus für Strings eines Computers finden Sie in natsort().

Detailliertere Informationen finden Sie auf Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) Seite.

Siehe auch sort(), natsort(), strnatcmp() und strnatcasecmp().

next (PHP 3, PHP 4 >= 4.0.0)

Rückt den internen Zeiger eines Arrays vor

mixed **next** (array array)

Diese Funktion gibt das nächste Element des Arrays zurück, auf das der interne Zeiger des Arrays gerade zeigt. Sind keine Elemente mehr vorhanden, wird FALSE zurückgegeben.

next() verhält sich ähnlich wie current(), mit einem Unterschied. Der interne Zeiger wird um ein Element vorgerückt, bevor das Element zurückgegeben wird, d.h. es wird das nächste Element zurückgegeben, und der interne Zeiger um eins erhöht. Zeigt der interne Zeiger hinter das Ende der Elementliste, wird FALSE zurückgegeben.

Warnung

Enthält das Array leere Elemente oder Elemente mit dem Schlüssel 0, wird für diese Elemente ebenfalls FALSE zurückgegeben. Um ein Array, das leere Elemente oder Elemente mit dem Schlüssel 0 enthalten könnte richtig durchzugehen, verwenden Sie die Funktion each().

Siehe auch current(), end(), prev() und reset().

pos (PHP 3, PHP 4 >= 4.0.0)

Liefert das aktuelle Element eines Arrays

mixed **pos** (array array)

Dies ist nur ein Alias für die Funktion current().

Siehe auch end(), next(), prev() und reset().

prev (PHP 3, PHP 4 >= 4.0.0)

Verkleinert den internen Zeiger eines Arrays

mixed **prev** (array array)

Liefert das Element vor der Position des Arrays, auf die der interne Zeiger gerade zeigt. Sind keine Elemente mehr vorhanden, wird FALSE zurückgegeben.

> **Warnung**
>
> Enthält das Array leere Elemente, wird für diese ebenfalls FALSE zurückgegeben. Um ein Array mit leeren Elementen richtig durchzugehen, verwenden Sie die Funktion each().

prev() verhält sich wie next(), nur dass hier der interne Zeiger um eine Position zurück, und nicht nach vorne gerückt wird.

Siehe auch current(), end(), next() und reset().

range (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Erstellt ein Array mit einem Bereich von Elementen

array **range** (mixed low, mixed high)

range() gibt ein Array mit Elementen im Bereich von *low* bis *high* zurück. Wenn low > high, wird die Sequenz von high nach low sein.

Beispiel 1. range()

```
foreach(range(0, 9) as $number) {
    echo $number;
}
foreach(range('a', 'z') as $letter) {
    echo $letter;
}
foreach(range('z', 'a') as $letter) {
    echo $letter;
}
```

Anmerkung: Vor der Version 4.1.0 hat die Funktion **range()** nur aufsteigende Integer Arrays erstellt. Die Unterstützung für Zeichenfolgen und Arrays in absteigender Reihenfolge wurde in 4.1.0 hinzugefügt.

Beispiel 2. Dekrementieren von Bereichen und Zeichenketten simulieren

```
# array_reverse kann dazu verwendet werden,
# die Reihenfolge eines Bereiches umzudrehen
foreach(array_reverse(range(0,9)) as $number) {
    echo $number;
}

# array_map() kann benutzt werden, um Integer-Werte mittels
# chr() in Zeichen umzuwandeln
foreach(array_map('chr', range(ord('a'),ord('z'))) as $character) {
    echo $character;
}
```

Siehe shuffle() für ein weiteres Anwendungsbeispiel.

reset (PHP 3, PHP 4 >= 4.0.0)

Setzt den internen Zeiger eines Arrays auf sein erstes Element

mixed **reset** (array array)

reset() setzt den internen Zeiger von `array` auf das erste Element zurück.

reset() gibt den Wert des ersten Arrayelements zurück.

Siehe auch current(), each(), next() und prev().

rsort (PHP 3, PHP 4 >= 4.0.0)

Sortiert ein Array in umgekehrter Reihenfolge

void **rsort** (array array [, int sort_flags])

Diese Funktion sortiert ein Array in umgekehrter Reihenfolge (vom höchsten zum niedrigsten Wert).

Beispiel 1. rsort()

```
$fruits = array ("Zitrone", "Orange", "Banane", "Apfel");
rsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

Dieses Beispiel würde folgendes ausgeben:

```
0 = Zitrone
1 = Orange
2 = Banane
3 = Apfel
```

Die Früchte wurden in umgekehrter alphabetischer Reihenfolge sortiert.

Sie können das Verhalten der Sortierung mittels dem optionalen Parameter *sort_flags* beeinflussen, für Details siehe sort().

Siehe auch arsort(), asort(), ksort(), sort() und usort().

shuffle (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Mischt die Elemente eines Arrays

void **shuffle** (array array)

Diese Funktion mischt die Reihenfolge der Elemente eines Arrays zufällig (shuffle). Benutzen Sie srand(), um den Startwert für den Zufallsgenerator festzulegen.

Beispiel 1. shuffle()

```
$numbers = range (1,20);
srand ((float)microtime()*1000000);
shuffle ($numbers);
while (list (, $number) = each ($numbers)) {
    echo "$number ";
}
```

Siehe auch arsort(), asort(), ksort(), rsort(), sort() und usort().

sizeof (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl Elemente einer Variable

int **sizeof** (mixed var)

Die Funktion **sizeof()** ist ein Alias für count().

Siehe auch count().

sort (PHP 3, PHP 4 >= 4.0.0)

Sortiert ein Array

void **sort** (array array [, int sort_flags])

Diese Funktion sortiert ein Array. Die Elemente werden aufsteigend vom niedrigsten zum höchsten Wert sortiert.

Beispiel 1. sort()

```
<?php

$fruits = array ("Zitrone", "Orange", "Banane", "Apfel");
sort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "fruits[".$key."] = ".$val."\n";
}

?>
```

Dieses Beispiel würde folgendes anzeigen:

```
fruits[0] = Apfel
fruits[1] = Banane
fruits[2] = Orange
fruits[3] = Zitrone
```

Die Früchte wurden in alphabetischer Reihenfolge sortiert.

Der optionale zweite Parameter *sort_flags* kann benutzt werden, um das Sortierverhalten mit den folgenden Flags zu beeinflussen.

Flags für Sortiertypen:

- SORT_REGULAR - Vergleiche Einträge normal
- SORT_NUMERIC - Vergleiche Einträge numerisch
- SORT_STRING - Vergleiche Einträge als Strings

Siehe auch arsort(), asort(), ksort(), natsort(), natcasesort(), rsort(), usort(), array_multisort() und uksort().

Anmerkung: Der zweite Parameter wurde in PHP 4 eingeführt.

uasort (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Sortiert ein Array mittels einer benutzerdefinierten Vergleichsfunktion und behält Indexassoziationen bei.

void **uasort** (array array, function cmp_function)

Diese Funktion sortiert ein Array so, dass die Beziehung der Indices zu den Arrayelementen beibehalten bleibt. Dies wird hauptsächlich bei assoziativen Arrays angewandt, wenn die aktuelle Reihenfolge der Arrayelemente signifikant ist. Die Vergleichsfunktion wird vom Benutzer definiert.

Anmerkung: Siehe usort() und uksort() für Beispiele von benutzerdefinierten Vergleichsfunktionen

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

Siehe auch usort(), uksort(), sort(), asort(), arsort(), ksort(), und rsort().

uksort (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Sortiert ein Array nach Schlüsseln mittels einer benutzerdefinierten Vergleichsfunktion.

void **uksort** (array array, function cmp_function)

Diese Funktion sortiert die Schlüssel eines Arrays mittels einer benutzerdefinierten Vergleichsfunktion. Soll das Array nach nicht trivialen Kriterien sortiert werden, sollten Sie diese Funktion verwenden.

Beispiel 1. uksort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (4 => "vier", 3 => "drei", 20 => "zwanzig", 10 => "zehn");

uksort ($a, "cmp");
```

```
while (list ($key, $value) = each ($a)) {
    echo "$key: $value\n";
}
```

Dieses Beispiel würde folgendes ausgeben:

```
20: zwanzig
10: zehn
4: vier
3: drei
```

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

Siehe auch usort(), uasort(), sort(), asort(), arsort(), ksort(), natsort() und rsort().

usort (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Sortiert ein Array nach Werten mittels einer benutzerdefinierten Vergleichsfunktion.

void **usort** (array array, string cmp_function)

Diese Funktion sortiert ein Array nach seinen Werten mittels einer benutzerdefinierten Vergleichsfunktion. Soll das Array nach nicht trivialen Kriterien sortiert werden, sollten Sie diese Funktion verwenden.

Die Vergleichsfunktion muss einen Integer zurückgeben, der kleiner, gleich, oder größer ist als `NULL`, wenn für das erste Argument auch angenommen wird, dass es kleiner, gleich, oder größer ist als das zweite. Sind zwei Elemente gleich, ist deren Reihenfolge nicht definiert.

Beispiel 1. usort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (3, 2, 5, 6, 1);

usort ($a, "cmp");

while (list ($key, $value) = each ($a)) {
    echo "$key: $value\n";
```

```
}
```

Dieses Beispiel würde folgendes anzeigen:

```
0: 6
1: 5
2: 3
3: 2
4: 1
```

Anmerkung: Offensichtlich würde rsort() für diesen einfachen Fall besser passen.

Beispiel 2. usort() Beispiel mit einem mehrdimensionalen Array

```
function cmp ($a, $b) {
    return strcmp($a["fruit"], $b["fruit"]);
}

$fruits[0]["fruit"] = "Zitronen";
$fruits[1]["fruit"] = "Äpfel";
$fruits[2]["fruit"] = "Trauben";

usort($fruits, "cmp");

while (list ($key, $value) = each ($fruits)) {
    echo "\$fruits[$key]: " . $value["fruit"] . "\n";
}
```

Wird ein mehrdimensionales Array sortiert, enthalten $a und $b Referenzen zu dem ersten Index des Arrays.

Dieses Beispiel würde folgendes anzeigen:

```
$fruits[0]: Äpfel
$fruits[1]: Trauben
$fruits[2]: Zitronen
```

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

Beispiel 3. usort() mit Verwendung einer Member Funktion eines Objektes

```
class TestObj {
    var $name;

    function TestObj($name)
    {
        $this->name = $name;
    }

    /* Dies ist die statische Vergleichsfunktion: */
    function cmp_obj($a, $b)
    {
        $al = strtolower($a->name);
        $bl = strtolower($b->name);
        if ($al == $bl) return 0;
        return ($al > $bl) ? +1 : -1;
    }
}

$a[] = new TestObj("c");
$a[] = new TestObj("b");
$a[] = new TestObj("d");

uasort($a, array ("TestObj", "cmp_obj"));

foreach ($a as $item) {
    print $item->name."\n";
}
```

Dieses Beispiel würde folgendes anzeigen:

```
b
c
d
```

Warnung

Die zugrundeliegende quicksort Funktion in machen C Bibliotheken (wie z.B. Solaris Systeme) könnte einen PHP Crash verursachen, wenn die Vergleichsfunktion keine gleichbleibenden Werte zurückgibt.

Siehe auch uasort(), uksort(), sort(), asort(), arsort(),ksort(), natsort() und rsort().

III. Aspell Funktionen [veraltet]

Die **aspell()** Funktionen erlauben es ein Wort auf korrekte Rechtschreibung zu prüfen und Alternativen anzubieten.

> **Anmerkung:** Aspell arbeitet nur mit ziemlich alten (bis .27.* oder so) Versionen der Aspell Bibliothek zusammen. Weder dieses Modul noch jene Versionen der Aspell Bibliothek werden nicht mehr länger unterstützt. Falls Sie eine Rechtschreibprüfung mit PHP benutzen wollen, verwenden Sie statt dessen die Pspell Funktionen. Diese nutzen die Pspell Bibliothek und arbeiten mit neueren Versionen von Aspell zusammen.

Sie benötigen die aspell-Bibliothek von: http://aspell.sourceforge.net/.

aspell_new (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Lädt ein neues Wörterbuch [veraltet]

int **aspell_new** (string master, string personal)

aspell_new() öffnet ein neues Wörterbuch und liefert eine Wörterbuchkennung zurück, die von anderen aspell-Funktionen gebraucht wird. Liefert `FALSE` bei einem Fehler.

Beispiel 1. aspell_new

```
$aspell_link = aspell_new("english");
```

aspell_check (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Überprüft ein Wort [veraltet]

boolean **aspell_check** (int Wörterbuchkennung, string Wort)

aspell_check() überprüft die Rechtschreibung eines Wortes und liefert `TRUE`, wenn das Wort korrekt ist, ansonsten `FALSE`.

Beispiel 1. aspell_check

```
$aspell_link = aspell_new("english");

if (aspell_check($aspell_link, "testt")) {
    echo "Gültige Schreibweise";
} else {
    echo "Bedaure, falsche Schreibweise";
}
```

aspell_check-raw (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Überprüft ein Wort genauso wie es übergeben wird [veraltet]

boolean **aspell_check_raw** (int Wörterbuchkennung, string Wort)

aspell_check_raw() überprüft die Rechtschreibung eines Wortes ohne die Groß/Kleinschreibung zu variieren oder das Wort in irgendeiner Weise zu kürzen. Liefert `TRUE`, wenn das Wort korrekt ist und sonst `FALSE`.

Beispiel 1. aspell_check_raw()

```
$aspell_link = aspell_new("english");

if (aspell_check_raw($aspell_link, "test")) {
    echo "Gültige Schreibweise";
} else {
    echo "Bedaure, falsche Schreibweise";
}
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Schlägt mögliche Schreibweisen vor [veraltet]

array **aspell_suggest** (int Wörterbuchkennung, string Wort)

aspell_suggest() liefert ein Array mit möglichen Schreibweisen eines übergebenen Wortes.

Beispiel 1. aspell_suggest()

```
$aspell_link = aspell_new("english");

if (!aspell_check($aspell_link, "test")) {
    $suggestions = aspell_suggest($aspell_link, "test");

    foreach ($suggestions as $suggestion) {
        echo "Mögliche Schreibweisen: $suggestion<br>\n";
    }
}
```

IV. Mathematische Funktionen mit beliebiger Genauigkeit

In PHP 4 sind diese Funktionen nur verfügbar, wenn PHP mit `--enable-bcmath` konfiguriert wurde. In PHP 3 sind diese Funktionen nur verfügbar, wenn PHP nicht mit `--disable-bcmath` konfiguriert wurde.

> **Anmerkung:** Aufgrund von Lizenzänderungen wird die BCMATH Bibliothek getrennt von der Standard PHP Quell-Distribution ausgeliefert. Das tar-gzippte Archiv können Sie sich unter der URL http://www.php.net/extra/number4.tar.gz herunterladen. Für mehr Informationen lesen Sie bitte die Datei `README.BCMATH` in der PHP Distribution.

bcadd (PHP 3, PHP 4 >= 4.0.0)

Addition zweier Zahlen beliebiger Genauigkeit.

string **bcadd** (string linker Operand, string rechter Operand [, int Genauigkeit])

Addiert den *linken Operanden* zu dem *rechten Operanden* und liefert die Summe als Zeichenkette. Der optionale Parameter *Genauigkeit* kann angegeben werden, um die Anzahl der Nachkommastellen im Ergebnis zu bestimmen.

Siehe auch bcsub().

bccomp (PHP 3, PHP 4 >= 4.0.0)

Vergleich zweier Zahlen beliebiger Genauigkeit.

int **bccomp** (string linker Operand, string rechter Operand [, int Genauigkeit])

Vergleicht den *linken Operanden* mit dem *rechten Operanden* und liefert das Ergebnis als integer. Der optionale Parameter *Genauigkeit* bestimmt die Anzahl der Nachkommastellen, die für den Vergleich ausgewertet werden. Der Rückgabewert ist 0, wenn die beiden Operanden gleich sind. Wenn der *linke Operand* größer ist als der *rechte Operand*, wird +1 zurückgeliefert; wenn der *linke Operand* kleiner ist als der *rechte Operand*, wird -1 zurückgeliefert.

bcdiv (PHP 3, PHP 4 >= 4.0.0)

Division zweier Zahlen beliebiger Genauigkeit.

string **bcdiv** (string linker Operand, string rechter Operand [, int Genauigkeit])

Dividiert den *linken Operanden* durch den *rechten Operanden*. Der optionale Parameter *Genauigkeit* legt die Anzahl der Nachkommastellen im Ergebnis fest.

Siehe auch bcmul().

bcmod (PHP 3, PHP 4 >= 4.0.0)

Modulo zweier Zahlen mit beliebiger Genauigkeit.

string **bcmod** (string linker Operand, string Modulus)

Liefert den *linken Operanden* modulo *Modulus*.

Siehe auch bcdiv().

bcmul (PHP 3, PHP 4 >= 4.0.0)

Multiplikation zweier Zahlen beliebiger Genauigkeit.

string **bcmul** (string linker Operand, string rechter Operand [, int Genauigkeit])

Multipliziert den `linken Operanden` mit dem `rechten Operanden`. Der optionale Parameter `Genauigkeit` legt die Anzahl der Nachkommastellen im Ergebnis fest.

Siehe auch bcdiv().

bcpow (PHP 3, PHP 4 >= 4.0.0)

Potenz mit beliebiger Genauigkeit.

string **bcpow** (string x, string y [, int Genauigkeit])

Potenziert `x` mit `y`. Der optionale Parameter `Genauigkeit` legt die Anzahl der Nachkommastellen im Ergebnis fest.

Siehe auch bcsqrt().

bcscale (PHP 3, PHP 4 >= 4.0.0)

Setzt die Genauigkeit aller BC math-Funktionen.

string **bcscale** (int Genauigkeit)

Diese Funktionen legt die Standard-Genauigkeit aller in der Folge aufgerufenden BC math-Funktionen fest, die nicht explizit die Genauigkeit durch den optionalen Parameter setzen.

bcsqrt (PHP 3, PHP 4 >= 4.0.0)

Quadratwurzel mit beliebiger Genauigkeit.

string **bcsqrt** (string Operand [, int Genauigkeit])

Liefert die Quadratwurzel von `Operand`. Der optionale Parameter `Genauigkeit` legt die Anzahl der Nachkommastellen im Ergebnis fest.

Siehe auch bcpow().

bcsub (PHP 3, PHP 4 >= 4.0.0)

Subtrahiert zwei Zahlen mit beliebiger Genauigkeit.

string **bcsub** (string linker Operand, string rechter Operand [, int Genauigkeit])

Subtrahiert den `rechten Operanden` vom `linken Operanden` und liefert das Ergebnis als Zeichenkette. Der optionale Parameter `Genauigkeit` legt die Anzahl der Nachkommastellen im Ergebnis fest.

Siehe auch bcadd().

V. Bzip2 Komprimierungsfunktionen

Dieses Module nutzt die Funktionen der bzip2 (http://sources.redhat.com/bzip2/)-Bibliothek von Julian Seward, um transparent bzip2 (.bz2)-komprimierte Dateien lesen und schreiben zu können.

Die Bzip2-Unterstützung in PHP ist nicht standardmäßig aktiviert. Es ist erforderlich, die --with-bz2-Option beim compilieren von PHP zu aktivieren. Dazu ist die bzip2/libbzip2 version >= 1.0.x. nötig.

Ein kleines Code-Beispiel

Dieses Beispiel öffnet eine tempräre Datei und schreibt einen Teststring in diese, dann gibt es den Inhalt der Datei aus.

Beispiel 1. Ein kleines bzip2() Beispiel

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";

// open file for writing
$bz = bzopen($filename, "w");

// write string to file
bzwrite($bz, $str);

// close file
bzclose($bz);

// open file for reading
$bz = bzopen($filename, "r");

// read 10 characters
print bzread($bz, 10);

// output until end of the file (or the next 1024 char) and close it.
print bzread($bz);

bzclose($bz);

?>
```

bzclose (PHP 4 >= 4.0.4)

Schließe einen bzip2-Dateizeiger

int **bzclose** (int bz)

Schließt die bzip2-Datei, die von dem Zeiger *bz* referenziert wird.

Gibt TRUE bei Erfolg und FALSE bei einem Fehler zurück.

Der Dateizeiger muß gültig sein, und auf eine Datei zeigen, die erfolgreich mittels der bzopen()-Funktion geöffnet wurde.

Siehe auch bzopen().

bzcompress (PHP 4 >= 4.0.4)

Komprimiert einen String in bzip2-encodierte Daten

string **bzcompress** (string source [, int blocksize [, int workfactor]])

bzcompress() komprimiert den *source*-String und gibt ihn als bzip2-komprimierte Daten zurück.

Der optionale Parameter *blocksize* spezifiziert die Blockgröße während der Komprimierung und darf zwischen 1 und 9 liegen. 9 ergibt dabei den besten Komprimierungsfaktor, benötigt jedoch auch die meisten Systemressourcen. Der Standardwert für *blocksize* ist 4.

Der optionale Parameter *workfactor* bestimmt, wie die Funktion sich im schlimmsten Fall, stark wiederholende Eingabedaten zu verhalten hat. Der Wer kann zwischen 0 und 250 liegen, 0 ist ein spezieller Fall, 30 ist der Standardwert. Unabhängig vom Parameter *workfactor* wird die generierte Ausgabe immer die selbe sein.

Beispiel 1. bzcompress() Beispiel

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

Siehe auch bzdecompress().

bzdecompress (PHP 4 >= 4.0.4)

Dekomprimiert bzip2-komprimierte Daten

string **bzdecompress** (string source [, int small])

bzdecompress() dekomprimiert den *source*-Parameter, der bzip2-komprimierte Daten enthält und gibt die dekomprimierten Daten zurück. Wenn der optionale Parameter *small* TRUE ist, wird ein

alternative Dekomprimierungsalgorithmus verwendet, der weniger Speicher benötigt, jedoch nur halb so schnell läuft. Weitere Informationen sind in der bzip2 Dokumentation (http://sources.redhat.com/bzip2/) zu finden.

Beispiel 1. bzdecompress()

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);

print( "Compressed String: " );
print( $bzstr );
print( "\n<br>n" );

$str = bzdecompress($bzstr);
print( "Decompressed String: " );
print( $str );
print( "\n<br>n" );
?>
```

Siehe auch bzcompress().

bzerrno (PHP 4 >= 4.0.4)

Gibt einen bzip2-Fehlercode zurück

int **bzerrno** (int bz)

Gibt den Fehlercode eines beliebigen Fehlercodes, der von dem Dateizeiger *bz* zurückgegeben wird.

Siehe auch bzerror() und bzerrstr().

bzerror (PHP 4 >= 4.0.4)

Gibt den bzip2 Fehlercode und eine Fehlerbeschreinung in einem Array zurück.

array **bzerror** (int bz)

Gibt den Fehlercode und die Fehlerbeschreibung eines beliebigen von dem Dateizeiger *bz* gelieferten bzip2-Fehlers in einem assoziativen Array zurück.

Beispiel 1. bzerror() Beispiel

```
<?php
$error = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
```

```
?>
```

Siehe auch bzerrno() und bzerrstr().

bzerrstr (PHP 4 >= 4.0.4)

Gibt eine bzip2-Fehlermeldung zurück

string **bzerrstr** (int bz)

Gibt eine Fehlermeldung eines bzip2-Fehler zurück, anhand des Dateizeigers *bz* identifiziert wird.

Siehe auch bzerrno() und bzerror().

bzflush (PHP 4 >= 4.0.4)

Erzwinge das schreiben aller gepufferten Daten

int **bzflush** (int bz)

Erzwingt ein Schreiben alle gepufferten bzip2-Daten für den Dateizeiger *bz*.

Gibt TRUE bei Erfolg, FALSE bei einem Fehler zurück.

Siehe auch bzread() und bzwrite().

bzopen (PHP 4 >= 4.0.4)

Öffnet eine bzip2-komprimierte Datei

int **bzopen** (string filename, string mode)

Öffnet eine bzip2 (.bz2)-Datei, um sie zu lesen oder zu schreiben. Der Parameter *filename* ist der Names der Datei, die geöffnet werden soll. Der *mode*-Parameter ist mit der fopen() Vergleich ('r' für lesenden Zugriff, 'w' für schreibenden Zugriff usw.).

Wenn das öffnen Fehlschlägt, gibt die Funktion FALSE zurück, andernfalls wird ein Zeiger auf die geöffnete Datei zurückgegeben.

Beispiel 1. bzopen() Beispiel

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$decompressed_file = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "The contents of /tmp/foo.bz2 are: " );
print( "\n<br>n" );
```

```
print( $decompressed_file );
?>
```

Siehe auch bzclose().

bzread (PHP 4 >= 4.0.4)

liest binäre Daten aus einer bzip2 komprimierten Datei

string **bzread** (int bz [, int length])

bzread() liest bis zu *length* Bytes aus dem bzip2-Dateizeiger *bz*. Hört mit dem Lesen auf, wenn *length* (dekomprimierte) Bytes gelesen wurden oder das Ende der Datei (EOF) erreicht ist, welcher Fall auch immer zuerst auftritt. Wenn der optionale Parameter *length* nicht angegeben wird, wird die Funktion **bzread()** 1024 unkomprimierte Bytes lesen.

Beispiel 1. bzread() Beispiel

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

Siehe auch bzwrite() und bzopen().

bzwrite (PHP 4 >= 4.0.4)

bzip2-Datei schreiben

int **bzwrite** (int bz, string data [, int length])

bzwrite() schreibt den Inhalt des Strings *data* in den bzip2-Dateistream, auf den *bz* zeigt. Wenn der optionale Parameter *length* übergeben wird, wird das Schreiben abgebrochen, nachdem *length* Bytes unkomprimierte Bytes geschrieben, oder das Ende des Strings erreicht wurde.

Beispiel 1. bzwrite() Beispiel

```
<?php
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

Siehe auch bzread() und bzopen().

VI. Kalender-Funktionen

Dieses Modul bietet eine Reihe von Funktionen, die die Umwandlung von Daten zwischen verschiedenen Kalendern erleichtern. Die gemeinsame Basis für diese Umwandlung bildet das Julianische Datum J. D. (nicht zu verwechseln mit dem Julianischen Kalender). Das J. D. entspricht der Anzahl der Tage seit dem 1. 1. 4713 v. Chr. Jede Umrechnung zwischen zwei beliebigen Kalendern erfordert den Zwischenschritt über das J. D.

Weitere Informationen zum Thema liefert die Seite http://genealogy.org/~scottlee/cal-overview.html, aus der Teile der Erklärungen in den folgenden Funktionsbeschreibungen stammen, sowie die sehr ausführliche Homepage zum Thema ZEIT (http://www.fortunecity.de/parkalleen/goldschmied/17/z1.htm).

Warnung

In PHP3 und älteren Versionen von PHP4 (bis einschließlich PHP4RC1) sind diese Funktionen nur verfügbar, wenn Sie die Kalender-Erweiterungen unter dl/calendar übersetzt haben. Weitere Informationen finden Sie in der Datei dl/README.

JDToGregorian (PHP 3, PHP 4 >= 4.0.0)

Konvertierung vom Julianischen Datum zum Gregorianischen Kalender

string **jdtogregorian** (int julianday)

Diese Funktion konvertiert den in *julianday* angegebenen Tag im Julianischen Datum in einen String, der das Datum nach dem Gregorianischen Kalender in der Form "MM/DD/YYYY" enthält.

GregorianToJD (PHP 3, PHP 4 >= 4.0.0)

Konvertierung vom Gregorianischen Kalender zum Julianischen Datum

int **gregoriantojd** (int month, int day, int year)

Diese Funktion konvertiert das in *month*, *day* und *year* übergebene Datum im Gregorianischen Kalender in einen Tag im Julianischen Datum.

Gültige Daten liegen im Bereich von 4714 v. Chr. bis 9999 n. Chr. Der Gregorianische Kalender wurde allerdings erst am 15. Oktober 1582 (bzw. am 5. Oktober nach dem bis dahin gültigen Julianischen Kalender) von Papst Gregor XIII. eingeführt, in einigen Ländern erst sehr viel später. So übernahmen die Briten den Gregorianischen Kalender 1752, die UDSSR 1918 und Griechenland erst im Jahre 1923. In den meisten europäischen Ländern wurde vor dem Gregorianischen der Julianische Kalender benutzt.

Beispiel 1. Calendar functions

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDToJulian (PHP 3, PHP 4 >= 4.0.0)

Konvertierung vom Julianischen Datum zum Julianischen Kalender

string **jdtojulian** (int julianday)

Diese Funktion konvertiert den in *julianday* angegebenen Tag im Julianischen Datum in einen String, der das entsprechnde Datum im Julianischen Kalender in der Form "MM/DD/YYYY" enthält.

JulianToJD (PHP 3, PHP 4 >= 4.0.0)

Konvertierung vom Julianischen Kalender zum Julianischen Datum

int **juliantojd** (int month, int day, int year)

Diese Funktion konvertiert das in `month`, `day` und `year` übergebene Datum im Gregorianischen Kalender in einen Tag im Julianischen Datum.

Gültige Daten liegen im Bereich von 4714 v.Chr. bis 9999 n.Chr. Der Julianische Kalender wurde allerdings erst 46 v.Chr. eingeführt, und die Details der Anwendung stabilisierten sich frühestens im Jahre 8 n. Chr., z. T. vielleicht sogar erst im vierten Jahrhundert nach Christus. Weiterhin betrachteten nicht alle Kulturen, die sich nach diesem Kalender richteten, den Januar als den ersten Monat des Jahres.

JDToJewish (PHP 3, PHP 4 >= 4.0.0)

Konvertierung vom Julianischen Datum zum Jüdischen Kalender

string **jdtojewish** (int julianday)

Diese Funktion konvertiert den in `julianday` angegebenen Tag im Julianischen Datum in einen String, der das entsprechnde Datum im Jüdischen Kalender in der Form "MM/DD/YYYY" enthält.

JewishToJD (PHP 3, PHP 4 >= 4.0.0)

Konvertiert vom Jüdischen Kalender zum Julianischen Datum

int **jewishtojd** (int month, int day, int year)

Diese Funktion konvertiert das in `month`, `day` und `year` übergebene Datum im Jüdischen Kalender in einen Tag im Julianischen Datum.

Gültige Daten beginnen mit dem Jahr 1 (3761 v. Chr.). Der Jüdische Kalender ist bereits seit mehreren Jahrtausenden im Gebrauch, wobei ein neuer Monat jeweils mit der ersten Beobachtung des Neumonds beginnt.

JDToFrench (PHP 3, PHP 4 >= 4.0.0)

Konvertiert ein Julianisches Datum zum Kalender der Französischen Revolution

string **jdtofrench** (int month, int day, int year)

Diese Funktion konvertiert den in `julianday` angegebenen Tag im Julianischen Datum in einen String, der das entsprechnde Datum im Kalender der Französischen Revolution in der Form "MM/DD/YYYY" enthält.

FrenchToJD (PHP 3, PHP 4 >= 4.0.0)

Konvertiert ein Datum der Französischen Revolution zu einem Julianischen Datum

int **frenchtojd** (int month, int day, int year)

Diese Funktion konvertiert das in `month`, `day` und `year` übergebene Datum im Kalender der Französischen Revolution in einen Tag im Julianischen Datum.

Diese Funktion verarbeitet nur Daten der Jahre 1 bis 14 (22. September 1792 bis 22. September 1806 im Gregorianischen Kalender), dieser Zeitraum geht aber immer noch über die Lebensdauer des Revolutionskalenders hinaus.

JDMonthName (PHP 3, PHP 4 >= 4.0.0)

Bestimmt den Monat aus dem Julianischen Datum

string **jdmonthname** (int julianday, int mode)

Diese Funktion bestimmt den Monatsnamen für den in `julianday` übergebenen Tag im Julianischen Datum. Die Ausgabe erfolgt als String in Abhängigkeit von `mode`.

Tabelle 1. Mögliche `mode` Werte

Mode	Bedeutung
0	Gregorianisch (Abk.)
1	Gregorianisch
2	Julianisch - (Abk.)
3	Julianisch
4	Jüdisch
5	Französisch revolutionär

JDDayOfWeek (PHP 3, PHP 4 >= 4.0.0)

Bestimmt den Wochentag aus einem Julianischen Datum

mixed **jddayofweek** (int julianday, int mode)

Diese Funktion bestimmt den Wochentag für den in `julianday` übergebenen Tag im Julianischen Datum. Die Ausgabe erfolgt in Abhängigkeit von `mode` als integer oder string.

Tabelle 1. Mögliche `mode` Werte

Mode	Bedeutung
0	Nummer des Tages als integer (0=Sontag, 1=Montag, ...)

Mode	Bedeutung
1	Tagesname als String (Englisch-Gregorianisch)
2	abgekürzter Tagesname als String (Englisch-Gregorianisch)

easter_date (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Zeitpunkt des Osterfestes (0 Uhr) als UNIX-Timestamp

int **easter_date** ([int year])

Diese Funktion liefert den UNIX-Timestamp (in Sekunden seit 1970) zurück, der dem Begin des Ostersonntag im durch *year* spezifizierten Jahr entspricht. Wird kein Jahr angegeben, so wird der Zeitpunkt für das laufende Jahr bestimmt.

Warnung

Liegt das angegebene Jahr vor 1970 oder nach 2037 (auf 32 bit-Systemen), so wird eine Warnung ausgegeben und das Ergebnis ist nicht gültig.

Beispiel 1. easter_date() Beispiel

```
echo date ("M-d-Y", easter_date(1999));        /* " 4. April 1999" */
echo date ("M-d-Y", easter_date(2000));        /* "23. April 2000" */
echo date ("M-d-Y", easter_date(2001));        /* "15. April 2001" */
```

Das Datum des Osterfestes wurde im Jahre 325 auf dem Konzil von Nicaea auf den ersten Sonntag nach Frühlingsvollmond, also dem Vollmond während oder direkt nach der Tagundnachtgleiche am 21. März, festgelegt. Der hier genutzte Algorithmus wurde um das Jahr 532 von Dionysius Exiguus eingeführt. Bis zum Jahr 1752 wird nach dem Julianischen Kalender ein einfacher 19jähriger Zyklus für die Bestimmung der Mondphasen benutzt. Im September 1752 übernahmen dann auch Britannien und seine Kolonien den von Clavius und Lilius vorgeschlagenen und im Oktober 1582 von Papst Gregor dem XIII. eingeführten Gregorianischen Kalender. Hierdurch wurden zwei Korrekturfaktoren eingeführt, um den Zyklus genauer abzubilden.

(Der Code basiert auf einen C-Programm von Simon Kershaw, <webmaster@ely.anglican.org>)

Siehe auch easter_days() für die Bestimmung von Osterfesten vor 1970 oder nach 2037.

easter_days (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Anzahl der Tage zwischen dem 21. März und Ostersonntag

int **easter_days** (int year)

Diese Funktion ermittelt den Abstand in Tagen zwischen der Frühlings-Tagundnachtgleiche und dem Ostersonntag im durch `year` angegebenen Jahr. Wird kein Jahr angegeben, so wird der Wert für das laufende Jahr bestimmt.

Diese Funktion kann im Gegensatz zu easter_date() das Osterfest auch in Jahren vor 1970 und nach 2037 bestimmen.

Beispiel 1. easter_date() Beispiel

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

Siehe auch easter_date().

unixtojd (PHP 4 >= 4.0.0)

Konvertiert UNIX-Timestamp in Julianisches Datum

int **unixtojd** ([int timestamp])

Diese Funktion konvertiert den in `timestamp` übergebenen Wert (in Sekunden seit dem 1.1.1970) in das entsprechende Julianische Datum. Wird kein `timestamp` übergeben, so wird das Julianische Datum des aktuellen Tages zurückgegeben.

Siehe auch jdtounix().

Anmerkung: Diese Funktion ist nur in PHP-Versionen nach PHP4RC1 verfügbar.

jdtounix (PHP 4 >= 4.0.0)

Konvertiert Julianisches Datum in UNIX-Timestamp

int **jdtounix** (int jday)

Diese Funktion liefert einen UNIX Timestamp (in Sekunden seit dem 1.1.1970) zurück, der dem in `jday` übergebenen Julianischen Datum entspricht. Liegt `jday` außerhalb des darstellbaren Zeitbereichs (Gregorianische Jahre von 1970 bis 2037 bzw. 2440588 <= `jday` <= 2465342), so wird FALSE zurückgegeben.

Siehe auch unixtojd().

Anmerkung: Diese Funktion ist nur in PHP-Versionen nach PHP4RC1 verfügbar.

VII. CCVS API Funktionen

Die Funktionsschnittstelle der CCVS API erlaubt es direkt von PHP Scripten aus mit CCVS zu arbeiten. CCVS ist RedHat's (http://www.redhat.com/) Lösung zu dem "Mann in der Mitte" bei der Abwicklung von Kreditkarten. Es stellt die Möglichkeit zur Verfügung sich direkt über einen Unixrechner und ein Modem zu einem Kartenanbieter zu verbinden. Mit hilfe des CCVS Modules für PHP kann man Kreditkarten direkt durch CCVS über PHP Scripte abwickeln. Die folgende Referenz wird diesen Vorgang umreissen.

Um CCVS Unterstützung zu aktivieren, sollte man erst das CCVS Installationsverzeichnis überprüfen. Es ist notwendig PHP mit der Option `--with-ccvs` zu configurieren. Wenn diese Option ohne den Pfad zu CCVS installation angegeben wird, versucht PHP es in den voreingestellten Pfaden einer CCVS standard installation (/usr/local/ccvs) zu finden. Wenn CCVS nicht den Standard Pfad hat, ist es nötig configure mit `--with-ccvs=$ccvs_pfad` zu starten, wobei $ccvs_pfad der Pfad zu installation ist. Anzumerken ist, das CCVS Unterstützung vorrausetzt das $ccvs_pfad/lib und $ccvs_pfad/include existieren, und die include cv_api.h im include Verzeichnis, und libccvs.a im lib Verzeichnis liegen.

Ausserdem muss ein ccvsd Prozess für die Einstellungen laufen die mit PHP Skripten arbeiten sollen. Es ist nötig das die PHP Prozesse unter dem selben Benutzer laufen für den auch ccvs installiert wurde (bsp: wenn ccvs unter dem user 'ccvs' installiert wurde, müssen die PHP Prozesse ebenfalls als 'ccvs' laufen.)

Weiter Informationen zu CCVS kann man unter http://www.redhat.com/products/ccvs finden.

An diesem Dokumentationsteil wird noch gearbeitet. RedHat unterhält eine geringfügig veraltete aber immernoch brauchbare Dokumentation unter:
http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

ccvs_init (PHP 4 >= 4.0.2)

Initialize CCVS for use

string **ccvs_init** (string name)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_done (PHP 4 >= 4.0.2)

Terminate CCVS engine and do cleanup work

string **ccvs_done** (string sess)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_new (PHP 4 >= 4.0.2)

Create a new, blank transaction

string **ccvs_new** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_add (PHP 4 >= 4.0.2)

Add data to a transaction

string **ccvs_add** (string session, string invoice, string argtype, string argval)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_delete (PHP 4 >= 4.0.2)

Delete a transaction

string **ccvs_delete** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_auth (PHP 4 >= 4.0.2)

Perform credit authorization test on a transaction

string **ccvs_auth** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_return (PHP 4 >= 4.0.2)

Transfer funds from the merchant to the credit card holder

string **ccvs_return** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_reverse (PHP 4 >= 4.0.2)

Perform a full reversal on an already-processed authorization

string **ccvs_reverse** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_sale (PHP 4 >= 4.0.2)

Transfer funds from the credit card holder to the merchant

string **ccvs_sale** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_void (PHP 4 >= 4.0.2)

Perform a full reversal on a completed transaction

string **ccvs_void** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_status (PHP 4 >= 4.0.2)

Check the status of an invoice

string **ccvs_status** (string session, string invoice)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_count (PHP 4 >= 4.0.2)

Find out how many transactions of a given type are stored in the system

int **ccvs_count** (string session, string type)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_lookup (PHP 4 >= 4.0.2)

Look up an item of a particular type in the database #

string **ccvs_lookup** (string session, string invoice, int inum)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_report (PHP 4 >= 4.0.2)

Return the status of the background communication process

string **ccvs_report** (string session, string type)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_command (PHP 4 >= 4.0.2)

Performs a command which is peculiar to a single protocol, and thus is not available in the general CCVS API

string **ccvs_command** (string session, string type, string argval)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ccvs_textvalue (PHP 4 >= 4.0.2)

Get text return value for previous function call

string **ccvs_textvalue** (string session)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

VIII. COM Support Funktionen für Windows

Das Component Object Model (COM) ist eine Technologie, welche die Wiederverwendung von Code erlaubt, egal in welcher Sprache dieser geschrieben wurde. Die Komponenten stellen Schnittstellen bereit, welche die verfügbaren Funktionen, aber nicht deren Implementierung bekanntgeben (z.B. auf welcher Maschine die Komponente gespeichert ist und die Executable, die sie enthält). Anwendungen können über die COM-API beliebige Schnittstellen erzeugen. Man sich das wie einen super Remote Procedure Call (RPC) Mechanismus mit einigen grundlegenden Objekten vorstellen. Es trennt die Implementation von der Schnittstelle.

COM unterstützt Versioning, die Trennung von Implementation und Schnittstelle, sowie das Verstecken von Implementierungsdetails, wie z.B. Position der Executable und die Sprache, in welcher die Komponente geschrieben wurde.

Die COM Funktionen sind nur in der Windowsversion von PHP verfügbar.

Weitere Informationen zu COM finden Sie in der COM specification (http://www.microsoft.com/Com/resources/comdocs.asp), oder in Don Box's Yet Another COM Library (YACL) (http://www.developmentor.com/dbox/yacl.htm).

COM (unknown)

COM Klasse

Synopsis

```
$obj = new COM("server.object")
```

Die COM Klasse bietet ein Gerüst zur Integration von (D)COM Komponenten in Ihre PHP Skripte.

string **COM::COM** (string module_name [, string server_name [, int codepage]])

COM Klassen Konstruktor. Parameter:

module_name

Name oder Klassen-ID der gewünschten Komponente.

server_name

Name des DCOM Servers, von welchem die Komponente geholt werden soll. Ist `NULL` angegeben, wird `localhost` angenommen. Um DCOM zuzulassen, muss `com.allow_dcom` in der `php.ini` auf `TRUE` gesetzt sein.

codepage

Spezifiziert die Codepage, welche zur Konvertierung der PHP-Strings in Unicode-Strings und umgekehrt verwendet wird. Mögliche Werte sind `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` und `CP_UTF8`.

Beispiel 1. COM Beispiel (1)

```
// Word starten
$word = new COM("word.application") or die("Konnte Word nicht instantiieren");
print "Word geladen, Version {$word->Version}\n";

//Fenster anzeigen
$word->Visible = 1;

//ein leeres Dokument öffnen
$word->Documents->Add();

//irgendetwas tun
$word->Selection->TypeText("Dies ist ein Test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//Word schließen
$word->Quit();

//Das Objekt freigeben
$word->Release();
```

```
$word = null;
```

Beispiel 2. COM example (2)

```
$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_columns; $i++)
{
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF)
{
    for ($i=0; $i < $num_columns; $i++)
    {
        echo $fld[$i]->value . "\t";
    }
    echo "\n";
    $rowcount++;            // increments rowcount
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs->Release();
$conn->Release();

$rs = null;
$conn = null;
```

VARIANT (unknown)

VARIANT Klasse

Synopsis

```
$vVar = new VARIANT($var)
```

Ein einfacher Container, um Variablen VARIANT Strukturen zu verpacken.

string **VARIANT::VARIANT** ([mixed value [, int type [, int codepage]]])

VARIANT Klassen-Konstruktor. Parameter:

value

Anfangswert. Ist er nicht angegeben, wird ein VT_EMPTY Objekt erzeugt.

type

spezifiziert den Typ vom Inhalt des VARIANT Objektes. Mögliche Werte sind `VT_UI1`, `VT_UI2`, `VT_UI4`, `VT_I1`, `VT_I2`, `VT_I4`, `VT_R4`, `VT_R8`, `VT_INT`, `VT_UINT`, `VT_BOOL`, `VT_ERROR`, `VT_CY`, `VT_DATE`, `VT_BSTR`, `VT_DECIMAL`, `VT_UNKNOWN`, `VT_DISPATCH` und `VT_VARIANT`. Diese Werte sind wechselseitig exklusiv, können aber mit `VT_BYREF` vereinigt werden, um zu spezifizieren, dass es sich um einen Wert handelt. Wird er weggelassen, wird der Typ von *`value`* verwendet. Konsultieren Sie bitte die MSDN Library für weitere Informationen.

codepage

gibt die Codepage an, welche zur Konvertierung der PHP-Strings in Unicode-Strings und umgekehrt verwendet wird. Mögliche Werte sind `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` und `CP_UTF8`.

com_load (PHP 3>= 3.0.3)

Erstellt eine neue Referenz auf eine COM Komponente

string **com_load** (string module name [, string server name [, int codepage]])

com_load() erstellt eine neue COM Komponente, und gibt eine Referenz darauf zurück. Im Fehlerfall wird `FALSE` zurückgegeben. Mögliche Werte für *`codepage`* sind `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7`, und `CP_UTF8`.

com_invoke (PHP 3>= 3.0.3)

Ruft eine Methode einer COM Komponente auf

mixed **com_invoke** (resource com_object, string function_name [, mixed function parameters, ...])

com_invoke() ruft eine Methode des mittels *`com_object`* referenzierten COM Komponente auf. Gibt bei Erfolg den Rückgabewert von *`function_name`*, oder im Fehlerfall `FALSE` zurück.

com_propget (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Liefert den Wert der Eigenschaft einer COM Komponente

mixed **com_propget** (resource com_object, string property)

Diese Funktion ist ein Alias von com_get().

com_get (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Liefert den Wert der Eigenschaft einer COM Komponente

mixed **com_get** (resource com_object, string property)

Liefert den Wert der Eigenschaft `property` der mittels `com_object` referenzierten COM Komponente. Gibt im Fehlerfall `FALSE` zurück.

com_propput (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Weist einer Eigenschaft einer COM Komponente einen Wert zu

void **com_propput** (resource com_object, string property, mixed value)

Diese Funktion ist ein Alias für com_set().

com_propset (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Weist einer Eigenschaft einer COM Komponente einen Wert zu

void **com_propset** (resource com_object, string property, mixed value)

Diese Funktion ist ein Alias für com_set().

com_set (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Weist einer Eigenschaft einer COM Komponente einen Wert zu.

void **com_set** (resource com_object, string property, mixed value)

Setzt den Wert der Eigenschaft `property` der COM Komponente `com_object`. Gibt bei Erfolg den neu gesetzten Wert, oder im Fehlerfall `FALSE` zurück.

com_addref (PHP 4 >= 4.1.0)

Erhöht den Referenzzähler der Komponente

void **com_addref** (void)

Erhöht den Referenzzähler der Komponente.

com_release (PHP 4 >= 4.1.0)

Reduziert den Referenzzähler der Komponente

void **com_release** (void)

Reduziert den Referenzzähler der Komponente.

com_isenum (PHP 4 >= 4.1.0)

Holt eine IEnumVariant

void **com_isenum** (object com_module)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

com_load_typelib (PHP 4 >= 4.1.0)

Lädt eine Typelib

void **com_load_typelib** (string typelib_name [, int case_insensitive])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

IX. Klassen- und Objekt-Funktionen

Einführung

About

Diese Funktionen ermöglichen Ihnen den Zugriff auf Informationen über Klassen und Objektinstanzen. Sie können den Namen der Klasse ermitteln, zu der ein Objekt gehört, sowie ihre Eigenschaften und -methoden. Mit diesen Funktionen können Sie nicht nur die Klassenmitgliedschaft eines Objektes herausfinden, sondern auch ihre Abstammung (d. h. welche Klasse die des Objektes erweitert).

Anwendungsbeispiel

In diesem Beispiel definieren wir zuerst eine Basisklasse und eine Erweiterung dieser Klasse. Die Basisklasse beschreibt ein beliebiges Gemüse, ob es eßbar ist oder nicht, sowie seine Farbe. Die Subklasse `Spinat` fügt eine Methode hinzu, um sie zu kochen und eine weitere, um herauszufinden, ob sie gekocht ist.

Beispiel 1. classes.inc

```
<?php

// Basisklasse mit Eigenschaften und Methoden
class Gemuese {

    var $essbar;
    var $farbe;

    function Gemuese( $essbar, $farbe="grün" ) {
        $this->essbar = $essbar;
        $this->farbe = $farbe;
    }

    function ist_essbar() {
        return $this->essbar;
    }

    function welche_farbe() {
        return $this->farbe;
    }

} // Ende der Klasse Gemuese


// erweitert die Basisklasse
class Spinat extends Gemuese {

    var $gekocht = false;
```

```
    function Spinat() {
        $this->Gemuese( true, "grün" );
    }

    function koche_es() {
        $this->gekocht = true;
    }

    function ist_gekocht() {
        return $this->gekocht;
    }

} // Ende der Klasse Spinat

?>
```

Jetzt instantiieren wir zwei Objekte von diesen Klassen und geben Informationen über sie aus, einschließlich ihrer Abstammung. Wir definieren auch einige Hilfsfunktionen, hauptsächlich um eine hübsche Ausgabe der Variablen zu erhalten.

Beispiel 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// utility functions

function zeige_vars($obj) {
    $arr = get_object_vars($obj);
    while (list($prop, $val) = each($arr))
        echo "\t$prop = $val\n";
}

function zeige_methoden($obj) {
    $arr = get_class_methods(get_class($obj));
    foreach ($arr as $method)
        echo "\tfunction $method()\n";
}

function klassen_abstammung($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
        echo "Objekt $obj gehört zur Klasse ".get_class($$obj);
        echo " einer Subklasse von $class\n";
    } else {
        echo "Object $obj gehört nicht zu einer Subklasse von $class\n";
    }
}

// Instantiiere zwei Objekte

$veggie = new Gemuese(true,"blau");
```

```
$leafy = new Spinat();

// Informationen über die Objekte ausgeben
echo "veggie: KLASSE ".get_class($veggie)."\n";
echo "leafy: KLASSE ".get_class($leafy);
echo ", ELTERN ".get_parent_class($leafy)."\n";

// Zeige Eigenschaften von veggie
echo "\nveggie: Eigenschaften\n";
zeige_vars($veggie);

// und Methoden von leafy
echo "\nleafy: Methoden\n";
zeige_methoden($leafy);

echo "\nAbstammung:\n";
klassen_abstammung("leafy", "Spinat");
klassen_abstammung("leafy", "Gemuese");
?>
</pre>
```

Wichtig ist in diesem Beispiel, daß das Objekt `$leafy` eine Instanz der Klasse `Spinat` ist, die eine Subklasse von `Gemuese` ist. Darum gibt der letzte Teil des obigen Skripts folgendes aus:

```
       [...]
Abstammung:
Objekt leafy gehört nicht zu einer Subklasse von Spinat
Objekt leafy gehört zur Klasse spinat einer Subklasse von Gemuese
```

call_user_method (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Aufruf einer benutzerdefinierten Methode eines bestimmten Objektes

mixed **call_user_method** (string method_name, object obj [, mixed parameter [, mixed ...]])

Ruft die durch *method_name* angegebene Methode des benutzerdefinierten Objektes *obj* auf. Ein Anwendungsbeispiel sehen Sie unten, wo wir eine Klasse definieren, ein Objekt instantiieren und **call_user_method()** benutzen, um seine Methode `zeige_info` indirekt aufzurufen.

```
<?php
class Land {
    var $NAME;
    var $TLD;

    function Land($name, $tld) {
        $this->NAME = $name;
        $this->TLD = $tld;
    }

    function zeige_info($prestr="") {
        echo $prestr."Land: ".$this->NAME."\n";
        echo $prestr."Top-Level-Domain: ".$this->TLD."\n";
    }
}

$cntry = new Land("Peru","pe");

echo "* Direkter Aufruf der Methode\n";
$cntry->zeige_info();

echo "\n* Indirekter Aufruf derselben Methode\n";
call_user_method ("zeige_info", $cntry, "\t");
?>
```

Siehe auch call_user_func().

class_exists (PHP 4 >= 4.0.0)

Prüft, ob eine Klasse definiert ist

bool **class_exists** (string class_name)

Diese Funktion ergibt `TRUE`, falls eine Klasse mit dem Namen *class_name* definiert ist, ansonsten `FALSE`.

get_class (PHP 4 >= 4.0.0)

Gibt den Namen der Klasse eines Objektes zurück

string **get_class** (object obj)

Diese Funktion gibt den Namen der Klasse zurück, von der *obj* eine Instanz ist.

Siehe auch get_parent_class(), is_subclass_of()

get_class_methods (PHP 4 >= 4.0.0)

Liefert die Namen aller Methoden einer Klasse

array **get_class_methods** (string class_name)

Diese Funktion liefert ein String-Array zurück, das mit den Namen aller in der Klasse *class_name* definerten Methoden gefüllt ist.

get_class_vars (PHP 4 >= 4.0.0)

Liefert die Standard-Elemente einer Klasse

array **get_class_vars** (string class_name)

Diese Funktion liefert ein String-Array zurück, das mit den Standard-Elementen der Klasse *class_name* gefüllt ist. Standard-Elemente sind all die Elemente, die bereits in der Klassendefinition mit 'var' definiert wurden.

get_declared_classes (PHP 4 >= 4.0.0)

Gibt ein Array mit den Namen der definierten Klassen zurück

array **get_declared_classes** (void)

Diese Funktion gibt ein Array mit den Namen aller im aktuellen Skript definierten Klassen zurück.

> **Anmerkung:** In PHP 4.0.1pl2 werden drei zusätzliche Klassen am Anfang des Array zurückgegeben: `stdClass` (definiert in `Zend/zend.c`), `OverloadedTestClass` (definiert in `ext/standard/basic_functions.c`) und `Directory` (definiert in `ext/standard/dir.c`).
>
> Abhängig von den Optionen, mit denen PHP kompiliert wurde, könnnen noch weitere Klassen vorhanden sein.

get_object_vars (PHP 4 >= 4.0.0)

Liefert die Elemente eines Objekts

array **get_object_vars** (object obj)

Diese Funktion gibt ein assoziatives Array der für das angegebene Objekt *obj* definierten Eigenschaften zurück. Wenn Variablen der Klasse, von der *obj* eine Instanz ist, kein Wert zugewiesen wurde, werden diese nicht im Array zurückgegeben.

Beispiel 1. Gebrauch von get_object_vars()

```
<?php
class Punkt2D {
    var $x, $y;
    var $bezeichnung;

    function Punkt2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setzeBezeichnung($bezeichnung) {
        $this->bezeichnung = $bezeichnung;
    }

    function gibPunkt() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "bezeichnung" => $this->bezeichnung);
    }
}

$p1 = new Punkt2D(1.233, 3.445);
print_r(get_object_vars($p1));
// "$bezeichnung" ist deklariert, aber nicht definiert
// Array
// (
//     [x] => 1.233
//     [y] => 3.445
// )

$p1->setzeBezeichnung("point #1");

print_r(get_object_vars($p1));
// Array
// (
//     [x] => 1.233
//     [y] => 3.445
//     [bezeichnung] => point #1
// )

?>
```

Siehe auch get_class_methods(), get_class_vars()

get_parent_class (PHP 4 >= 4.0.0)

Gibt den Namen der Elternklasse eines Objektes zurück

string **get_parent_class** (object obj)

Diese Funktion gibt den Namen der Elternklasse der Klasse zurück, von der *obj* eine Instanz ist.

Siehe auch get_class(), is_subclass_of()

is_subclass_of (PHP 4 >= 4.0.0)

Bestimmt, ob ein Objekt zu einer Subklasse der angegebenen Klasse gehört

bool **is_subclass_of** (object obj, string superclass)

Diese Funktion ergibt TRUE, falls das Objekt *obj* zu einer Subklasse von *superclass* gehört, ansonsten FALSE.

Siehe auch get_class(), get_parent_class()

method_exists (PHP 4 >= 4.0.0)

Prüft, ob Methode in einer Klasse definiert ist

bool **method_exists** (object object, string method_name)

Diese Funktion liefert TRUE, wenn eine Methode mit Namen *method_name* für das in *object* übergebene Objekt definiert wurde, sonst FALSE.

Siehe auch function_exists()

X. ClibPDF Funktionen

Mit der ClibPDF können Sie PDF-Dokumente mit PHP erstellen. Diese Bibliothek ist zum Herunterladen erhältlich von FastIO (http://www.fastio.com/), aber für deren kommerzielle Verwendung ist es erforderlich, dass Sie eine Lizenz erwerben. Funktionalität und API der ClibPDF sind der PDFlib ähnlich.

Diese Dokumentation sollten Sie zusammen mit dem ClibPDF-Handbuch lesen, weil letzteres diese Bibliothek viel ausführlicher erklärt.

Die Namen vieler Funktionen in der ClibPDF, im PHP-Modul und in der PDFlib sind gleich. Außer cpdf_open() nehmen alle Funktionen die Bezeichnung des Dokuments als ihren ersten Parameter.

Diese Bezeichnung wird gegenwärtig nicht intern verwendet, weil ClibPDF die gleichzeitige Erzeugung mehrerer PDF-Dokumente nicht unterstützt. Da das Ergebnis nicht vorhersehbar ist, sollten Sie es am besten nicht einmal testen. Die Folgen, die das in einer multi-threaded Umgebung hätte, kann ich nicht abschätzen. Laut ClibPDF-Autor wird sich das in einer der nächsten Versionen ändern (als dieses geschrieben wurde, war Version 1.10 aktuell). Verwenden Sie das pdflib-Modul, wenn Sie diese Funktionalität benötigen.

> **Anmerkung:** Die Funktion cpdf_set_font() hat sich seit PHP 3 geändert und unterstützt nun asiatische Zeichensätze. Der Kodierungsparameter ist keine Zahl mehr, sondern eine Zeichenkette.

Eine nette Eigenschaft der ClibPDF (und der PDFlib) ist die Möglichkeit, das PDF-Dokument vollständig im Speicher zu erzeugen, ohne temporäre Dateien zu verwenden. Sie bietet auch die Möglichkeit, Koordinaten in einer vordefinierten Längeneinheit zu übergeben. Dies ist eine praktische Fähigkeit, die bei Verwendung der PDFlib aber auch mit pdf_translate() simuliert werden kann.

Eine weitere nette Eigenschaft von ClibPDF ist die Tatsache, dass jede Seite jederzeit geändert werden kann, selbst wenn schon eine neue Seite geöffnet wurde. Die Funktion cpdf_set_current_page() erlaubt es, die aktuelle Seite zu verlassen und mit den Änderungen einer anderen Seite fortzufahren.

Die meisten Funktionen sind ziemlich einfach zu benutzen. Der schwierigste Teil ist wahrscheinlich die Erzeugung eines sehr einfachen PDF-Dokuments überhaupt. Das folgende Beispiel sollte für Sie eine gute Starthilfe sein. Es erzeugt ein Dokument mit einer Seite. Die Seite enthält den Umriss des Textes "Times-Roman" in 30-Punkt-Schrift. Der Text ist unterstrichen.

Beispiel 1. Ein einfaches ClibPDF-Beispiel

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Seite 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
```

```
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

Die pdflib-Distribution enthält ein komplexeres Beispiel, das eine Reihe von Seiten mit einer Analoguhr erzeugt. Hier ist dieses Beispiel, das unter Verwendung der ClibPDF-Erweiterung in PHP konvertiert wurde:

Beispiel 2. pdfclock-Beispiel der pdflib 2.0 Distribution

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $mar-
gin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */

  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* minute strokes */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha &lt; 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* 5 minute strokes */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha &lt; 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }
```

```
  $ltime = getdate();

  /* draw hour hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw minute hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw second hand */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* draw little circle at center */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Fügt eine Anmerkung ein

void **cpdf_add_annotation** (int PDF-Dokument, float ulx, float uly, float orx, float ory, string Titel, string Inhalt [, int Modus])

Die Funktion **cpdf_add_annotation()** fügt eine Anmerkung ein, bei der die untere linke Ecke bei (*ulx*, *uly*) und die obere rechte Ecke bei (*orx*, *ory*) liegt.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, dil wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Fügt ein Lesezeichen für die aktuelle Seite hinzu

void **cpdf_add_outline** (int PDF-Dokument, string Text)

Die Funktion **cpdf_add_outline()** fügt ein Lesezeichen mit dem Text *Text* hinzu, das auf die aktuelle Seite zeigt.

Beispiel 1. Hinzufügen einer Seitenübersicht

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Seite 1");
// ...
// etwas Zeichnung
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_arc (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Zeichnet einen Kreisbogen

void **cpdf_arc** (int PDF-Dokument, float x-koor, float y-koor, float Radius, float Anfang, float Ende [, int Modus])

Die Funktion **cpdf_arc()** zeichnet einen Kreisbogen mit dem Mittelpunkt (*x-koor*, *y-koor*) und dem Radius *Radius*, der mit dem Winkel *Anfang* beginnt und mit dem Winkel *Ende* endet.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_circle().

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Beginnt einen Textabschnitt

void **cpdf_begin_text** (int PDF-Dokument)

Die Funktion **cpdf_begin_text()** beginnt einen Textabschnitt. Er muss mit cpdf_end_text() beendet werden.

Beispiel 1. Ausgabe von Text

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Etwas Text");
cpdf_end_text($pdf)
?>
```

Siehe auch: cpdf_end_text().

cpdf_circle (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Zeichnet einen Kreis

void **cpdf_circle** (int PDF-Dokument, float x-koor, float y-koor, float Radius [, int Modus])

Die Funktion **cpdf_circle()** zeichnet einen Kreis mit dem Mittelpunkt (*x-koor*, *y-koor*) und dem Radius *Radius*.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_arc().

cpdf_clip (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Beschränkt alle Zeichnungen auf den aktuellen Pfad

void **cpdf_clip** (int PDF-Dokument)

Die Funktion **cpdf_clip()** beschränkt alle Zeichnungen auf den aktuellen Pfad.

cpdf_close (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Schließt ein PDF-Dokument

void **cpdf_close** (int PDF-Dokument)

Die Funktion **cpdf_close()** schließt das PDF-Dokument. Sie sollte als letzte, sogar nach cpdf_finalize(), cpdf_output_buffer() und cpdf_save_to_file(), aufgerufen werden.

Siehe auch: cpdf_open().

cpdf_closepath (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Schließt den aktuellen Pfad

void **cpdf_closepath** (int PDF-Dokument)

Die Funktion **cpdf_closepath()** schließt den aktuellen Pfad.

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Schließt, füllt und zeichnet den aktuellen Pfad

void **cpdf_closepath_fill_stroke** (int PDF-Dokument)

Die Funktion **cpdf_closepath_fill_stroke()** schließt den aktuellen Pfad, füllt seinen Innenraum mit der aktuellen Füllfarbe und zeichnet ihn.

Siehe auch: cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill() und cpdf_setrgbcolor().

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Schließt einen Pfad und zeichnet eine Linie entlang des Pfades

void **cpdf_closepath_stroke** (int PDF-Dokument)

Die Funktion **cpdf_closepath_stroke()** ist eine Kombination aus cpdf_closepath() und cpdf_stroke(). Der Pfad wird geschlossen.

Siehe auch: cpdf_closepath() und cpdf_stroke().

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gibt den Text in der nächsten Zeile aus

void **cpdf_continue_text** (int PDF-Dokument, string Text)

Die Funktion **cpdf_continue_text()** gibt die Zeichenkette in *Text* in der nächsten Zeile aus.

Siehe auch: cpdf_show_xy(), cpdf_text(), cpdf_set_leading() und cpdf_set_text_pos().

cpdf_curveto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Zeichnet eine Kurve

void **cpdf_curveto** (int PDF-Dokument, float x1, float y1, float x2, float y2, float x3, float y3 [, int Modus])

Die Funktion **cpdf_curveto()** zeichnet eine Bezier-Kurve vom aktuellen Punkt zum Punkt (*x3*, *y3*), wobei (*x1*, *y1*) und (*x2*, *y2*) als Kontrollpunkte verwendet werden.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_moveto(), cpdf_rmoveto(), cpdf_rlineto() und cpdf_lineto().

cpdf_end_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Beendet einen Textabschnitt

void **cpdf_end_text** (int PDF-Dokument)

Die Funktion **cpdf_end_text()** beendet einen Textabschnitt, der mit cpdf_begin_text() begonnen wurde.

Beispiel 1. Ausgabe von Text

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Etwas Text");
cpdf_end_text($pdf)
?>
```

Siehe auch: cpdf_begin_text().

cpdf_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Füllt den aktuellen Pfad

void **cpdf_fill** (int PDF-Dokument)

Die Funktion **cpdf_fill()** füllt den Innenraum des aktuellen Pfades mit der aktuellen Füllfarbe.

Siehe auch: cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill() und cpdf_setrgbcolor().

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Füllt und zeichnet den aktuellen Pfad

void **cpdf_fill_stroke** (int PDF-Dokument)

Die Funktion **cpdf_fill_stroke()** füllt den Innenraum des aktuellen Pfades mit der aktuellen Füllfarbe und zeichnet den aktuellen Pfad.

Siehe auch: cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill() und cpdf_setrgbcolor().

cpdf_finalize (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Beendet ein Dokument

void **cpdf_finalize** (int PDF-Dokument)

Die Funktion **cpdf_finalize()** beendet das Dokument *`PDF-Dokument`*. Sie müssen danach trotzdem noch cpdf_close() aufrufen.

Siehe auch: cpdf_close().

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Beendet eine Seite

void **cpdf_finalize_page** (int PDF-Dokument, int Seitennummer)

Die Funktion **cpdf_finalize_page()** beendet die Seite mit der Seitennummer *`page number`*.

Diese Funktion hat nur den Zweck, Speicherplatz zu sparen. Eine abgeschlossene Seite belegt weniger Speicherplatz, kann aber nicht mehr verändert werden.

Siehe auch: cpdf_page_init().

cpdf_global_set_document_limits (PHP 4 >= 4.0.0)

Setzt Beschränkungen für alle PDF-Dokumente

void **cpdf_global_set_document_limits** (int maxSeiten, int maxSchriftarten, int maxAbbildungen, int maxBemerkungen, int maxObjekte)

Die Funktion **cpdf_global_set_document_limits()** setzt verschiedene Beschränkungen für Dokumente. Damit diese Funktion wirksam wird, muss sie vor cpdf_open() aufgerufen werden. Sie setzt die Beschränkungen für alle Dokumente, die danach geöffnet werden.

Siehe auch: cpdf_open().

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Öffnet ein JPEG-Bild

int **cpdf_import_jpeg** (int PDF-Dokument, string Dateiname, float x-koor, float y-koor, float Winkel, float Breite, float Höhe, float x-Skalierung, float y-Skalierung [, int Modus])

Die Funktion **cpdf_import_jpeg()** öffnet ein Bild, das unter dem Namen *Dateiname* gespeichert wurde. Das Bild muß im JPEG-Format vorliegen. Es wird auf der aktuellen Seite an der Position (*x-koor*, *y-koor*) platziert. Das Bild wird um *Winkel* Grad gedreht.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_place_inline_image().

cpdf_lineto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Zeichnet eine Linie

void **cpdf_lineto** (int PDF-Dokument, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_lineto()** zeichnet eine Linie vom aktuellen Punkt zum Punkt mit den Koordinaten (*x-koor*, *y-koor*).

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_moveto(), cpdf_rmoveto() und cpdf_curveto().

cpdf_moveto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die aktuelle Position fest

void **cpdf_moveto** (int PDF-Dokument, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_moveto()** setzt die aktuelle Position auf die Koordinaten *x-koor* und *y-koor*.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_rmoveto().

cpdf_newpath (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Beginnt einen neuen Pfad

void **cpdf_newpath** (int PDF-Dokument)

Die Funktion **cpdf_newpath()** beginnt auf dem durch den Parameter *PDF-Dokument* angegebenen Dokument einen neuen Pfad.

cpdf_open (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Öffnet ein neues PDF-Dokument

int **cpdf_open** (int Komprimierung [, string Dateiname])

Die Funktion **cpdf_open()** öffnet ein neues PDF-Dokument. Wenn *Komprimierung* ungleich 0 ist, schaltet dieser Parameter die Komprimierung des Dokuments ein. Der optionale Parameter *Dateiname* bestimmt die Datei in die das Dokument geschrieben wird. Wenn er nicht angegeben wurde, wird das Dokument im Speicher erzeugt und kann entweder mit cpdf_save_to_file() in eine Datei geschrieben werden oder mit cpdf_output_buffer() an die Standardausgabe geschickt werden.

Anmerkung: Der Rückgabewert wird in künftigen ClibPDF-Versionen als erster Parameter aller anderen Funktionen benötigt, die in ein PDF-Dokument schreiben.

Die ClibPDF-Bibliothek betrachtet den Dateinamen "-" als Synonym für stdout. Das funktioniert allerdings nicht, wenn PHP als Apache-Modul kompiliert wurde, weil die Art und Weise, in der ClibPDF die Ausgabe an stdout vornimmt, nicht mit Apache funktioniert. Sie können dieses Problem lösen, indem Sie keinen Dateinamen angeben und stattdessen cpdf_output_buffer() verwenden, um das PDF-Dokument auszugeben.

Siehe auch: cpdf_close() und cpdf_output_buffer().

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Gibt das PDF-Dokument aus dem Zwischenspeicher aus

void **cpdf_output_buffer** (int PDF-Dokument)

Die Funktion **cpdf_output_buffer()** gibt das PDF-Dokument an stdout aus. Das Dokument muss im Speicher erstellt worden sein, was der Fall ist, wenn cpdf_open() ohne den Parameter Dateiname aufgerufen wurde.

Siehe auch: cpdf_open().

cpdf_page_init (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Beginnt eine neue Seite

void **cpdf_page_init** (int PDF-Dokument, int Seitennummer, int Ausrichtung, float Höhe, float Breite [, float Einheit])

Die Funktion **cpdf_page_init()** beginnt eine neue Seite mit der Höhe *Höhe* und der Breite *Breite*. Die Seite hat die Nummer *Seitennummer* und die Ausrichtung *Ausrichtung*. *Ausrichtung* kann 0 für Hochformat oder 1 für Querformat sein. Der optionale Parameter *Einheit* gibt die Einheit des Koordinatensystems in Anzahl von Postscript-Punkten pro Einheit an. Da ein Zoll 72 Punkten entspricht, würde ein Wert von 72 die Einheit auf ein Zoll setzen. Auch der Standardwert ist 72.

Siehe auch: cpdf_set_current_page().

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Plaziert ein Bild auf der Seite

void **cpdf_place_inline_image** (int PDF-Dokument, int Bild, float x-koor, float y-koor, float Winkel, float Breite, float Höhe [, int Modus])

Die Funktion **cpdf_place_inline_image()** plaziert auf der Seite ein Bild, das mit der PHP-Image-Funktion erstellt wurde, an der Position (*x-koor*, *y-koor*). Das Bild kann gleichzeitig skaliert werden.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_import_jpeg().

cpdf_rect (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Zeichnet ein Rechteck

void **cpdf_rect** (int PDF-Dokument, float x-koor, float y-koor, float Breite, float Höhe [, int Modus])

Die Funktion **cpdf_rect()** zeichnet ein Rechteck, bei dem die linke untere Ecke im Punkt (*x-koor*, *y-koor*) liegt. Die Breite wird auf *Breite* und die Höhe auf *Höhe* gesetzt.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

cpdf_restore (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Stellt die zuvor gesicherte Umgebung wieder her

void **cpdf_restore** (int PDF-Dokument)

Die Funktion **cpdf_restore()** stellt die mit cpdf_save() gesicherte Umgebung wieder her. Sie funktioniert wie der Postscript-Befehl grestore und ist sehr nützlich, wenn Sie ein Objekt verschieben oder drehen wollen, ohne daß es sich auf andere Objekte auswirkt.

Beispiel 1. Sichern/Wiederherstellen

```
<?php
cpdf_save($pdf);
// nehmen Sie alle möglichen Arten von Drehungen, Verformungen,... vor
cpdf_restore($pdf)
?>
```

Siehe auch: cpdf_save().

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Zeichnet eine Linie (relativ)

void **cpdf_rlineto** (int PDF-Dokument, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_rlineto()** zeichnet eine Linie vom aktuellen Punkt zum relativen Punkt mit den Koordinaten (*x-koor*, *y-koor*).

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_moveto(), cpdf_rmoveto() und cpdf_curveto().

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Legt die relative aktuelle Position fest

void **cpdf_rmoveto** (int PDF-Dokument, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_rmoveto()** setzt die aktuelle Position relativ zu den Koordinaten *x-koor* und *y-koor*.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_moveto().

cpdf_rotate (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt den Drehwinkel fest

void **cpdf_rotate** (int PDF-Dokument, float Winkel)

Die Funktion **cpdf_rotate()** setzt die Drehung in Grad auf *Winkel*.

cpdf_rotate_text (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Legt den Drehwinkel für Text fest

void **cpdf_rotate_text** (int PDF-Dokument, float Winkel)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cpdf_save (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sichert die aktuelle Umgebung

void **cpdf_save** (int PDF-Dokument)

Die Funktion **cpdf_save()** sichert die aktuelle Umgebung. Sie funktioniert wie der Postscript-Befehl gsave und ist sehr nützlich, wenn Sie ein Objekt verschieben oder drehen wollen, ohne daß es sich auf andere Objekte auswirkt.

Siehe auch: cpdf_restore().

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Schreibt ein PDF-Dokument in eine Datei

void **cpdf_save_to_file** (int PDF-Dokument, string Dateiname)

Die Funktion **cpdf_save_to_file()** schreibt das PDF-Dokument in eine Datei, sofern es im Speicher erstellt wurde.

Diese Funktion wird nicht benötigt, wenn das PDF-Dokument unter Angabe eines Dateinamens als Parameter von cpdf_open() geöffnet wurde.

Siehe auch: cpdf_output_buffer() und cpdf_open().

cpdf_scale (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die Vergrößerung fest

void **cpdf_scale** (int PDF-Dokument, float x-Vergrößerung, float y-Vergrößerung)

Die Funktion **cpdf_scale()** setzt den Vergrößerungsfaktor in beide Richtungen.

cpdf_set_action_url (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Setzt einen Hyperlink

void **cpdf_set_action_url** (int PDF-Dokument, float xll, float yll, float xur, float xur, string URL [, int Modus])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt den Zeichenabstand fest

void **cpdf_set_char_spacing** (int PDF-Dokument, float Abstand)

Die Funktion **cpdf_set_char_spacing()** legt den Abstand zwischen den Zeichen fest.

Siehe auch: cpdf_set_word_spacing() cpdf_set_leading().

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt das Urheberfeld des PDF-Dokuments

void **cpdf_set_creator** (string Urheber)

Die Funktion **cpdf_set_creator()** setzt den Urheber eines PDF-Dokuments.

Siehe auch: cpdf_set_subject(), cpdf_set_title() und cpdf_set_keywords().

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Setzt die aktuelle Seite

void **cpdf_set_current_page** (int PDF-Dokument, int Seitennummer)

Die Funktion **cpdf_set_current_page()** setzt die Seite, an der alle Aktionen durchgeführt werden. Sie können solange zwischen den Seiten wechseln, bis Sie eine Seite mit cpdf_finalize_page() abschließen.

Siehe auch: cpdf_finalize_page().

cpdf_set_font (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Wählt die aktuelle Schriftart und -größe

void **cpdf_set_font** (int PDF-Dokument, string Schriftart, float Größe, string Kodierung)

Die Funktion **cpdf_set_font()** setzt die aktuelle Schriftart, Schriftgröße und -kodierung. Gegenwärtig werden nur Standard-Postscript-Schriftarten unterstützt.

Der Parameter *`Kodierung`* kann die folgenden Werte akzeptieren: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding" und "`NULL`". "`NULL`" steht für die schriftarteigene Kodierung.

Weitere Informationen, insbesondere wie asiatische Schriftarten unterstützt werden, finden Sie im ClibPDF-Handbuch.

cpdf_set_font_directories (PHP 4 >= 4.0.6)

Legt die Verzeichnisse fest, die bei Verwendung externer Schriftarten durchsucht werden

void **cpdf_set_font_directories** (int PDF-Dokument, string pfmdir, string pfbdir)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cpdf_set_font_map_file (PHP 4 >= 4.0.6)

Legt bei Verwendung externer Schriftarten die Datei mit den Schriftart-Dateiname-Zuordnungen fest

void **cpdf_set_font_map_file** (int PDF-Dokument, string Datei)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt die horizontale Textskalierung

void **cpdf_set_horiz_scaling** (int PDF-Dokument, float Maßstab)

Die Funktion **cpdf_set_horiz_scaling()** setzt die horizontale Skalierung auf *Maßstab* Prozent.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt das Schlüsselwortfeld des PDF-Dokuments

void **cpdf_set_keywords** (string Schlüsselwörter)

Die Funktion **cpdf_set_keywords()** setzt die Schlüsselwörter eines PDF-Dokuments.

Siehe auch: cpdf_set_title(), cpdf_set_creator() und cpdf_set_subject().

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt den Zeilenabstand

void **cpdf_set leading** (int PDF-Dokument, float Abstand)

Die Funktion **cpdf_set_leading()** setzt den Zeilenabstand. Dieser wird verwendet, wenn der Text durch cpdf_continue_text() ausgegeben wird.

Siehe auch: cpdf_continue_text().

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Legt den Übergang zwischen den Seiten fest

void **cpdf_set_page_animation** (int PDF-Dokument, int Übergang, float Dauer)

Die Funktion **cpdf_set_page_animation()** legt den Übergang zwischen aufeinanderfolgenden Seiten fest.

Übergang kann folgende Werte haben:

0 für keinen,
1 für zwei über den Bildschirm sausende Zeilen, die die nächste Seite enthüllen,
2 für mehrere über den Bildschirm sausende Zeilen, die die nächste Seite enthüllen,
3 für eine Box, die die nächste Seite enthüllt,
4 für eine einzelne über den Bildschirm sausende Zeile, die die nächste Seite enthüllt,
5, daß dich die alte Seite auflöst und die nächste Seite enthüllt,
6, daß der Auflösungseffekt vom einen Bildschirmrand zum anderen wandert,
7, daß die alte Seite einfach durch die neue Seite ersetzt wird (Grundeinstellung)

Der Wert *Dauer* ist die Anzahl der Sekunden während des Seitenwechsels.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt das Themenfeld des PDF-Dokuments

void **cpdf_set_subject** (string Thema)

Die Funktion **cpdf_set_subject()** setzt das Thema eines PDF-Dokuments.

Siehe auch: cpdf_set_title(), cpdf_set_creator() und cpdf_set_keywords().

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die Text-Matrix fest

void **cpdf_set_text_matrix** (int PDF-Dokument, array Matrix)

Die Funktion **cpdf_set_text_matrix()** legt die Matrix fest, die eine auf die aktuelle Schriftart angewandte Transformation beschreibt.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die Textposition fest

void **cpdf_set_text_pos** (int PDF-Dokument, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_set_text_pos()** legt die Position des Textes für den nächsten Aufruf von cpdf_show() fest.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Siehe auch: cpdf_show() cpdf_text().

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt fest, wie Text wiedergegeben wird

void **cpdf_set_text_rendering** (int PDF-Dokument, int Modus)

Die Funktion **cpdf_set_text_rendering()** legt fest, wie der Text wiedergegeben wird.

Die möglichen Werte für `Modus` sind 0=Text ausfüllen, 1=Text durchstreichen, 2=Text ausfüllen und durchstreichen, 3=unsichtbar, 4=Text ausfüllen und dem Begrenzungspfad hinzufügen, 5=Text durchstreichen und dem Begrenzungspfad hinzufügen, 6=Text ausfüllen und durchstreichen und dem Begrenzungspfad hinzufügen, 7=dem Begrenzungspfad hinzufügen.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die vertikale Textverschiebung fest

void **cpdf_set_text_rise** (int PDF-Dokument, float Verschiebung)

Die Funktion **cpdf_set_text_rise()** legt die Verschiebung des Texts um `Verschiebung` Einheiten nach oben fest.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt das Titelfeld PDF-Dokuments

void **cpdf_set_title** (string Titel)

Die Funktion **cpdf_set_title()** setzt den Titel eines PDF-Dokuments.

Siehe auch: cpdf_set_subject(), cpdf_set_creator() und cpdf_set_keywords().

cpdf_set_viewer_preferences (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Legt fest, wie das Dokument im Betrachter dargestellt wird

void **cpdf_set_viewer_preferences** (int PDF-Dokument, array Einstellungen)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt den Abstand zwischen den Wörtern fest

void **cpdf_set_word_spacing** (int PDF-Dokument, float Abstand)

Die Funktion **cpdf_set_word_spacing()** legt den Abstand zwischen den Wörtern fest.

Siehe auch: cpdf_set_char_spacing() cpdf_set_leading().

cpdf_setdash (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt eine gestrichelte Linie

void **cpdf_setdash** (int PDF-Dokument, float schwarz, float weiß)

Die Funktion **cpdf_setdash()** setzt das Muster einer gestrichelten Linie auf *weiß* weiße Einheiten und *schwarz* schwarze Einheiten. Wenn beide 0 sind, wird die Linie durchgezogen.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die Flachheit fest

void **cpdf_setflat** (int PDF-Dokument, float Flachheit)

Die Funktion **cpdf_setflat()** setzt die Flachheit auf einen Wert zwischen 0 und 100.

cpdf_setgray (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt Zeichen- und Füllfarbe auf einen Grauwert

void **cpdf_setgray** (int PDF-Dokument, float Grauwert)

Die Funktion **cpdf_setgray()** setzt die aktuelle Zeichen- und Füllfarbe auf den angegebenen Grauwert.

Siehe auch: cpdf_setrgbcolor_stroke() und cpdf_setrgbcolor_fill().

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt die Füllfarbe auf einen Grauwert

void **cpdf_setgray_fill** (int PDF-Dokument, float Grauwert)

The **cpdf_setgray_fill()** function sets the current gray value to fill a path.

Siehe auch: cpdf_setrgbcolor_fill().

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt die Zeichenfarbe auf einen Grauwert

void **cpdf_setgray_stroke** (int PDF-Dokument, float Grauwert)

Die Funktion **cpdf_setgray_stroke()** setzt die aktuelle Zeichenfarbe auf den angegebenen Grauwert.

Siehe auch: cpdf_setrgbcolor_stroke().

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt den Typ der Linienenden fest

void **cpdf_setlinecap** (int PDF-Dokument, int Typ)

Die Funktion **cpdf_setlinecap()** setzt den Typ der Linienenden auf einen Wert zwischen 0 und 2. 0 = stumpfes Ende, 1 = rund, 2 = vorspringendes Quadrat.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt den Typ der Linienverbindungen fest

void **cpdf_setlinejoin** (int PDF-Dokument, long Typ)

Die Funktion **cpdf_setlinejoin()** setzt den Typ der Linienverbindungen auf einen Wert zwischen 0 und 2. 0 = Gehrung, 1 = rund, 2 = schräg.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die Linienbreite fest

void **cpdf_setlinewidth** (int PDF-Dokument, float Breite)

Die Funktion **cpdf_setlinewidth()** setzt die Linienbreite auf *`Breite`*.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt die Gehrungsbegrenzung fest

void **cpdf_setmiterlimit** (int PDF-Dokument, float Wert)

Die Funktion **cpdf_setmiterlimit()** setzt die Gehrungsbegrenzung auf einen Wert größer oder gleich 1.

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt Zeichen- und Füllfarbe auf einen RGB-Farbwert

void **cpdf_setrgbcolor** (int PDF-Dokument, float Rotwert, float Grünwert, float Blauwert)

Die Funktion **cpdf_setrgbcolor()** setzt die aktuelle Zeichen- und Füllfarbe auf den angegebenen RGB-Farbwert.

Siehe auch: cpdf_setrgbcolor_stroke() und cpdf_setrgbcolor_fill().

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt die Füllfarbe auf einen RGB-Farbwert

void **cpdf_setrgbcolor_fill** (int PDF-Dokument, float Rotwert, float Grünwert, float Blauwert)

The **cpdf_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

Siehe auch: cpdf_setrgbcolor_stroke() und cpdf_setrgbcolor().

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt die Zeichenfarbe auf einen RGB-Farbwert

void **cpdf_setrgbcolor_stroke** (int PDF-Dokument, float Rotwert, float Grünwert, float Blauwert)

Die Funktion **cpdf_setrgbcolor_stroke()** setzt die aktuelle Zeichenfarbe auf den angegebenen RGB-Farbwert.

Siehe auch: cpdf_setrgbcolor_fill() und cpdf_setrgbcolor().

cpdf_show (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gibt Text an der aktuellen Position aus

void **cpdf_show** (int PDF-Dokument, string Text)

Die Funktion **cpdf_show()** gibt die Zeichenkette *`Text`* an der aktuellen Position aus.

Siehe auch: cpdf_text(), cpdf_begin_text() und cpdf_end_text().

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gibt Text an der angegebenen Position aus

void **cpdf_show_xy** (int PDF-Dokument, string Text, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_show_xy()** gibt die Zeichenkette *Text* an der Position mit den Koordinaten (*x-koor*, *y-koor*) aus.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

Anmerkung: Ohne die optionalen Parameter ist die Funktion **cpdf_show_xy()** mit cpdf_text() identisch.

Siehe auch: cpdf_text().

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Liefert die Breite einer Zeichenkette in der aktuellen Schriftart

float **cpdf_stringwidth** (int PDF-Dokument, string Text)

Die Funktion **cpdf_stringwidth()** gibt die Breite der Zeichenkette in *Text* zurück. Zuvor muß eine Schriftart festgelegt werden.

Siehe auch: cpdf_set_font().

cpdf_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Zeichnet eine Linie entlang des aktuellen Pfades

void **cpdf_stroke** (int PDF-Dokument)

Die Funktion **cpdf_stroke()** Zeichnet eine Linie entlang des aktuellen Pfades.

Siehe auch: cpdf_closepath() und cpdf_closepath_stroke().

cpdf_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gibt den Text mit Parametern aus

void **cpdf_text** (int PDF-Dokument, string Text, float x-koor, float y-koor [, int Modus [, float Richtung [, int Ausrichtungsmodus]]])

Die Funktion **cpdf_text()** gibt die Zeichenkette *Text* an der Position mit den Koordinaten (*x-koor*, *y-koor*) aus.

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen. Der

optionale Parameter *Richtung* bestimmt die Drehung des Texts in Grad. Der optionale Parameter *Ausrichtungsmodus* bestimmt, wie der Text ausgerichtet wird.

Die möglichen Werte können Sie der ClibPDF-Dokumentation entnehmen.

Siehe auch: cpdf_show_xy().

cpdf_translate (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Legt den Ursprung des Koordinatensystems fest

void **cpdf_translate** (int PDF-Dokument, float x-koor, float y-koor [, int Modus])

Die Funktion **cpdf_translate()** setzt den Ursprung des Koordinatensystems auf den Punkt (*x-koor*, *y-koor*).

Der optionale Parameter *Modus* bestimmt die Länge einer Einheit. Wenn er 0 ist oder weggelassen wurde, wird die Einheit verwendet, die für die Seite als Standard angegeben wurde. Andernfalls werden die Koordinaten unabhängig von der aktuellen Einheit in Postscript-Punkten gemessen.

XI. Crack Funktionen

Diese Funktionen erlauben die Nutzung der CrackLib Bibliothek, die dazu verwendet werden kann, die 'Stärke' eines Passworts zu überprüfen. Um diese Funktionen nutzen zu können, muß PHP mittels `--with-crack[=DIR]` mit Crack-Support compiliert werden. option.

Weiterführende Informationen zur CrackLib können unter http://www.users.dircon.co.uk/~crypto/ gefunden werden.

Die CrackLib ist hilfreich, um die 'Stärke' eines Passworts zu testen. Es werden Länge und die Benutzung von Gross- und Kleinschreibung überprüft. Zusätzlich wird das Passwort gegen ein CrackLib Wörterbuch überprüft. Die CrackLib liefert des weiteren hilfreiche Diagnosemeldungen, die helfen können, das Passwort 'stärker' zu machen.

Diese Beispiel zeigt, wie ein CrackLib Wörterbuch geöffnet wird, ein übergebenes Passwort überprüft wird und Diagnosemeldungen ausgewertet werden. Anschließend wird das Wörterbuch wieder geschlossen.

Beispiel 1. CrackLib Beispiel

```
<?php
// Öffne das CrackLib Wörterbuch
$dictionary = crack_opendict('/usr/local/lib/pw_dict')
     or die('Kann das CrackLib Wörterbuch nicht öffnen');

// Führe eine Überprüfung des Passworts durch
$check = crack_check($dictionary, 'gx9A2s0x');

// Empfange Rückmeldungen
$diag = crack_getlastmessage();
echo $diag; // 'starkes Passwort'

// Schliesse das Wörterbuch
crack_closedict($dictionary);
?>
```

Anmerkung: Wenn crack_check() `TRUE` zurückliefert, wird crack_getlastmessage() 'strong password' zurückliefern.

crack_opendict (PHP 4 >= 4.0.5)

Öffnet ein CrackLib Wörterbuch

resource **crack_opendict** (string dictionary)

Gibt bei Erfolg einen Zeiger auf das Wörterbuch zurück oder FALSE bei einem Fehler.

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

crack_opendict() öffnet das mit `dictionary` angegebene Wörterbuch, um es mit crack_check() zu benutzen.

> **Anmerkung:** Es kann nur ein Wörterbuch zur gleichen Zeit geöffnet sein.

Siehe auch: crack_check(), und crack_closedict().

crack_closedict (PHP 4 >= 4.0.5)

Schließt ein CrackLib Wörterbuch

bool **crack_closedict** ([resource dictionary])

Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

crack_closedict() schliesst den in `dictionary` angegebenen Zeiger. Wenn `dictionary` nicht genauer spezifiziert wird, wird das aktuell verwendetet Wörterbuch geschlossen.

crack_check (PHP 4 >= 4.0.5)

Führt eine Überprüfung des übergebenen Passwortes durch.

bool **crack_check** ([resource dictionary, string password])

Gibt TRUE zurück, wenn `password` 'stark' ist. Wenn nicht, wird FALSE zurückgegeben.

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

crack_check() führt eine Überprüfung des mit `password` übergebenen Strings über das mit `dictionary` angegebene Wörterbuch durch. Wenn `dictionary` nicht spezifiziert wird, wird das zuletzt geöffnete Wörterbuch genutzt.

crack_getlastmessage (PHP 4 >= 4.0.5)

Gibt die Nachricht der letzten Überprüfung zurück.

string **crack_getlastmessage** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

crack_getlastmessage() gibt die Nachricht der letzten Passwortüberprüfung zurück.

XII. CURL, Client URL Library Functions

PHP supports libcurl, a library created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form based upload, proxies, cookies, and user+password authentication.

In order to use the CURL functions you need to install the CURL (http://curl.haxx.se/) package. PHP requires that you use CURL 7.0.2-beta or higher. PHP will not work with any version of CURL below version 7.0.2-beta.

To use PHP's CURL support you must also compile PHP `--with-curl[=DIR]` where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the easy.h and curl.h files. There should be a file named "libcurl.a" located in the "lib" directory.

These functions have been added in PHP 4.0.2.

Once you've compiled PHP with CURL support, you can begin using the curl functions. The basic idea behind the CURL functions is that you initialize a CURL session using the curl_init(), then you can set all your options for the transfer via the curl_exec() and then you finish off your session using the curl_close(). Here is an example that uses the CURL functions to fetch the PHP homepage into a file:

Beispiel 1. Using PHP's CURL module to fetch the PHP homepage

```
<?php

$ch = curl_init ("http://www.php.net/");
$fp = fopen ("php_homepage.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_init (PHP 4 >= 4.0.2)

Initialize a CURL session

int **curl_init** ([string url])

The **curl_init()** will initialize a new session and return a CURL handle for use with the curl_setopt(), curl_exec(), and curl_close() functions. If the optional *url* parameter is supplied then the CURLOPT_URL option will be set to the value of the parameter. You can manually set this using the curl_setopt() function.

Beispiel 1. Initializing a new CURL session and fetching a webpage

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.zend.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

See also: curl_close(), curl_setopt()

curl_setopt (PHP 4 >= 4.0.2)

Set an option for a CURL transfer

bool **curl_setopt** (int ch, string option, mixed value)

The **curl_setopt()** function will set options for a CURL session identified by the *ch* parameter. The *option* parameter is the option you want to set, and the *value* is the value of the option given by the *option*.

The *value* should be a long for the following options (specified in the *option* parameter):

- *CURLOPT_INFILESIZE*: When you are uploading a file to a remote site, this option should be used to tell PHP what the expected size of the infile will be.
- *CURLOPT_VERBOSE*: Set this option to a non-zero value if you want CURL to report everything that is happening.
- *CURLOPT_HEADER*: Set this option to a non-zero value if you want the header to be included in the output.
- *CURLOPT_NOPROGRESS*: Set this option to a non-zero value if you don't want PHP to display a progress meter for CURL transfers

Anmerkung: PHP automatically sets this option to a non-zero parameter, this should only be changed for debugging purposes.

- `CURLOPT_NOBODY`: Set this option to a non-zero value if you don't want the body included with the output.
- `CURLOPT_FAILONERROR`: Set this option to a non-zero value if you want PHP to fail silently if the HTTP code returned is greater than 300. The default behavior is to return the page normally, ignoring the code.
- `CURLOPT_UPLOAD`: Set this option to a non-zero value if you want PHP to prepare for an upload.
- `CURLOPT_POST`: Set this option to a non-zero value if you want PHP to do a regular HTTP POST. This POST is a normal application/x-www-form-urlencoded kind, most commonly used by HTML forms.
- `CURLOPT_FTPLISTONLY`: Set this option to a non-zero value and PHP will just list the names of an FTP directory.
- `CURLOPT_FTPAPPEND`: Set this option to a non-zero value and PHP will append to the remote file instead of overwriting it.
- `CURLOPT_NETRC`: Set this option to a non-zero value and PHP will scan your ~./netrc file to find your username and password for the remote site that you're establishing a connection with.
- `CURLOPT_FOLLOWLOCATION`: Set this option to a non-zero value to follow any "Location: " header that the server sends as a part of the HTTP header (note this is recursive, PHP will follow as many "Location: " headers that it is sent.)
- `CURLOPT_PUT`: Set this option a non-zero value to HTTP PUT a file. The file to PUT must be set with the CURLOPT_INFILE and CURLOPT_INFILESIZE.
- `CURLOPT_MUTE`: Set this option to a non-zero value and PHP will be completely silent with regards to the CURL functions.
- `CURLOPT_TIMEOUT`: Pass a long as a parameter that contains the maximum time, in seconds, that you'll allow the curl functions to take.
- `CURLOPT_LOW_SPEED_LIMIT`: Pass a long as a parameter that contains the transfer speed in bytes per second that the transfer should be below during CURLOPT_LOW_SPEED_TIME seconds for PHP to consider it too slow and abort.
- `CURLOPT_LOW_SPEED_TIME`: Pass a long as a parameter that contains the time in seconds that the transfer should be below the CURLOPT_LOW_SPEED_LIMIT for PHP to consider it too slow and abort.
- `CURLOPT_RESUME_FROM`: Pass a long as a parameter that contains the offset, in bytes, that you want the transfer to start from.
- `CURLOPT_SSLVERSION`: Pass a long as a parameter that contains the SSL version (2 or 3) to use. By default PHP will try and determine this by itself, although, in some cases you must set this manually.
- `CURLOPT_SSL_VERIFYHOST`: Pass a long if cURL should verify the Common name of the peer certificate in the SSL handshake. A value of 1 denotes that we should check for the existence of the common name, a value of 2 denotes that we should make sure it matches the provided hostname.

- `CURLOPT_TIMECONDITION`: Pass a long as a parameter that defines how the CURLOPT_TIMEVALUE is treated. You can set this parameter to TIMECOND_IFMODSINCE or TIMECOND_ISUNMODSINCE. This is a HTTP-only feature.
- `CURLOPT_TIMEVALUE`: Pass a long as a parameter that is the time in seconds since January 1st, 1970. The time will be used as specified by the CURLOPT_TIMEVALUE option, or by default the TIMECOND_IFMODSINCE will be used.
- `CURLOPT_RETURNTRANSFER`: Pass a non-zero value if you want cURL to directly return the transfer instead of printing it out directly.

The `value` parameter should be a string for the following values of the `option` parameter:

- `CURLOPT_URL`: This is the URL that you want PHP to fetch. You can also set this option when initializing a session with the curl_init() function.
- `CURLOPT_USERPWD`: Pass a string formatted in the [username]:[password] manner, for PHP to use for the connection. connection.
- `CURLOPT_PROXYUSERPWD`: Pass a string formatted in the [username]:[password] format for connection to the HTTP proxy.
- `CURLOPT_RANGE`: Pass the specified range you want. It should be in the "X-Y" format, where X or Y may be left out. The HTTP transfers also support several intervals, separated with commas as in X-Y,N-M.
- `CURLOPT_POSTFIELDS`: Pass a string containing the full data to post in an HTTP "POST" operation.
- `CURLOPT_REFERER`: Pass a string containing the "referer" header to be used in an HTTP request.
- `CURLOPT_USERAGENT`: Pass a string containing the "user-agent" header to be used in an HTTP request.
- `CURLOPT_FTPPORT`: Pass a string containing the which will be used to get the IP address to use for the ftp "POST" instruction. The POST instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a hostname, a network interface name (under UNIX), or just a plain '-' to use the systems default IP address.
- `CURLOPT_COOKIE`: Pass a string containing the content of the cookie to be set in the HTTP header.
- `CURLOPT_SSLCERT`: Pass a string containing the filename of PEM formatted certificate.
- `CURLOPT_SSLCERTPASSWD`: Pass a string containing the password required to use the CURLOPT_SSLCERT certificate.
- `CURLOPT_COOKIEFILE`: Pass a string containing the name of the file containing the cookie data. The cookie file can be in Netscape format, or just plain HTTP-style headers dumped into a file.
- `CURLOPT_CUSTOMREQUEST`: Pass a string to be used instead of `GET` or `HEAD` when doing an HTTP request. This is useful for doing `DELETE` or other, more obscure, HTTP requests. Valid values are things like `GET`, `POST`, and so on; i.e. do not enter a whole HTTP request line here. For instance, entering 'GET /index.html HTTP/1.0\r\n\r\n' would be incorrect.

Anmerkung: Don't do this without making sure your server supports the command first.

- `CURLOPT_PROXY`: Give the name of the HTTP proxy to tunnel requests through.
- `CURLOPT_INTERFACE`: Pass the name of the outgoing network interface to use. This can be an interface name, an IP address or a host name.
- `CURLOPT_KRB4LEVEL`: Pass the KRB4 (Kerberos 4) security level. This anyone of the following strings (in order from least powerful, to most powerful): 'clear', 'safe', 'confidential', 'private'. If the string does not match one of these, then 'private' is used. If you set this to NULL, this disables KB4 security. KB4 security only works with FTP transactions currently.
- `CURLOPT_HTTPHEADER`: Pass an array of HTTP header fields to set.
- `CURLOPT_QUOTE`: Pass an array of FTP commands to perform on the server prior to the FTP request.
- `CURLOPT_POSTQUOTE`: Pass an array of FTP commands to execute on the server, after the FTP request has been performed.

The following options expect a file descriptor that is obtained by using the fopen() function:

- `CURLOPT_FILE`: The file where the output of your transfer should be placed, the default is STDOUT.
- `CURLOPT_INFILE`: The file where the input of your transfer comes from.
- `CURLOPT_WRITEHEADER`: The file to write the header part of the output into.
- `CURLOPT_STDERR`: The file to write errors to instead of stderr.

curl_exec (PHP 4 >= 4.0.2)

Perform a CURL session

bool **curl_exec** (int ch)

This function is should be called after you initialize a CURL session and all the options for the session are set. Its purpose is simply to execute the predefined CURL session (given by the `ch`).

> **Tipp:** Bei allem, dessen Ausgabe des Ergebnisses direkt im Browser angezeigt wird, können Sie die Funktionen zur Ausgabesteuerung verwenden. Damit lässt sich die Ausgabe dieser Funktion erfassen, und - zum Beispiel - in einem string speichern.

curl_close (PHP 4 >= 4.0.2)

Close a CURL session

void **curl_close** (int ch)

This function closes a CURL session and frees all resources. The CURL handle, `ch`, is also deleted.

curl_version (PHP 4 >= 4.0.2)

Return the current CURL version

string **curl_version** (void)

The **curl_version()** function returns a string containing the current CURL version.

curl_errno (PHP 4 >= 4.0.3)

Return an integer containing the last error number

int **curl_errno** (int ch)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

curl_error (PHP 4 >= 4.0.3)

Return a string contain the last error for the current session

string **curl_error** (int ch)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

curl_getinfo (PHP 4 >= 4.0.4)

Get information regarding a specific transfer

string **curl_getinfo** (int ch, int opt)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XIII. Cybercash Bezahl Funktionen

Diese Funktionen sind nur dann verfügbar, wenn der Interpreter mit der Option `--with-cybercash=[DIR]` kompiliert wurde, und sie wurden mit PHP 4 eingeführt.

cybercash_encr (PHP 4 >= 4.0.0)

???

array **cybercash_encr** (string wmk, string sk, string inbuff)

Diese Funktion gibt ein assoziatives Array mit den Elementen "errcode" und, falls "errcode" den Wert FALSE hat, "outbuff" (string), "outLth" (long) und "macbuff" (string) zurück.

cybercash_decr (PHP 4 >= 4.0.0)

???

array **cybercash_decr** (string wmk, string sk, string inbuff)

Diese Funktion gibt ein assoziatives Array mit den Elementen "errcode" und, falls "errcode" den Wert FALSE hat, "outbuff" (string), "outLth" (long) und "macbuff" (string) zurück.

cybercash_base64_encode (PHP 4 >= 4.0.0)

???

string **cybercash_base64_encode** (string inbuff)

cybercash_base64_decode (PHP 4 >= 4.0.0)

string **cybercash_base64_decode** (string inbuff)

XIV. Crédit Mutuel CyberMUT functions

This extension allows you to process credit cards transactions using Crédit Mutuel CyberMUT system (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CyberMUT is a popular Web Payment Service in France, provided by the Crédit Mutuel bank. If you are foreign in France, these functions will not be useful for you.

These functions are only available if PHP has been compiled with the `--with-cybermut[=DIR]` option, where DIR is the location of `libcm-mac.a` and `cm-mac.h`. You will require the appropriate SDK for your platform, which may be sent to you after your CyberMUT's subscription (contact them via Web, or go to the nearest Crédit Mutuel).

The use of these functions is almost identical to the original functions, except for the parameters of return for cybermut_creerformulairecm() and cybermut_creerreponsecm(), which are returned directly by functions PHP, whereas they had passed in reference in the original functions.

These functions have been added in PHP 4.0.6.

> **Anmerkung:** These functions only provide a link to CyberMUT SDK. Be sure to read the CyberMUT Developers Guide for full details of the required parameters.

cybermut_creerformulairecm (PHP 4 >= 4.0.5)

Generate HTML form of request for payment

string **cybermut_creerformulairecm** (string url_CM, string version, string TPE, string montant, string ref_commande, string texte_libre, string url_retour, string url_retour_ok, string url_retour_err, string langue, string code_societe, string texte_bouton)

cybermut_creerformulairecm() is used to generate the HTML form of request for payment.

Beispiel 1. First step of payment (equiv cgi1.c)

```
<?php
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

  $retour =  cybermut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");

  echo $retour;
?>
```

See also cybermut_testmac() and cybermut_creerreponsecm().

cybermut_testmac (PHP 4 >= 4.0.5)

Make sure that there no was data diddling contained in the received message of confirmation

bool **cybermut_testmac** (string code_MAC, string version, string TPE, string cdate, string montant, string ref_commande, string texte_libre, string code-retour)

cybermut_testmac() is used to make sure that there was not data diddling contained in the received message of confirmation. Pay attention to parameters `code-retour` and `texte-libre`, which cannot be evaluated as is, because of the dash. You must retrieve them by using:

```
<?php
```

```
  $code_retour=$HTTP_GET_VARS["code-retour"];
  $texte_libre=$HTTP_GET_VARS["texte-libre"];
?>
```

Beispiel 1. Last step of payment (equiv cgi2.c)

```
<?php
// Make sure that Enable Track Vars is ON.
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

$texte_libre = $HTTP_GET_VARS["texte-libre"];
$code_retour = $HTTP_GET_VARS["code-retour"];

$mac_ok = cybermut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$coc

if ($mac_ok) {

  //
  // insert data processing here
  //
  //

  $result=cybermut_creerreponsecm("OK");
} else {
  $result=cybermut_creerreponsecm("Document Falsifie");
}

?>
```

See also cybermut_creerformulairecm() and cybermut_creerreponsecm().

cybermut_creerreponsecm (PHP 4 >= 4.0.5)

Generate the acknowledgement of delivery of the confirmation of payment

string **cybermut_creerreponsecm** (string phrase)

cybermut_creerreponsecm() returns a string containing delivery acknowledgement message.

The parameter is "OK" if the message of confirmation of the payment was correctly identified by cybermut_testmac(). Any other chain is regarded as an error message.

See also cybermut_creerformulairecm() and cybermut_testmac().

XV. Cyrus IMAP administration functions

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cyrus_connect (PHP 4 >= 4.1.0)

Connect to a Cyrus IMAP server

resource **cyrus_connect** ([string host [, string port [, int flags]]])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cyrus_authenticate (PHP 4 >= 4.1.0)

Authenticate agaings a Cyrus IMAP server

bool **cyrus_authenticate** (resource connection [, string mechlist [, string service [, string user [, int minssf [, int maxssf]]]]])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cyrus_bind (PHP 4 >= 4.1.0)

Bind callbacks to a Cyrus IMAP connection

bool **cyrus_bind** (resource connection, array callbacks)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cyrus_unbind (PHP 4 >= 4.1.0)

Unbind ...

bool **cyrus_unbind** (resource connection, string trigger_name)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cyrus_query (PHP 4 >= 4.1.0)

Send a query to a Cyrus IMAP server

bool **cyrus_query** (resource connection, string query)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

cyrus_close (PHP 4 >= 4.1.0)

Close connection to a cyrus server

bool **cyrus_close** (resource connection)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XVI. Character type functions

The functions provided by this extension check whether a character or string falls into a certain character class according to the current locale (see also setlocale()).

When called with an integer argument these functions behave exactly like their C counterparts from "ctype.h".

When called with a string argument they will check every character in the string and will only return `TRUE` if every character in the string matches the requested criteria.

Passing anything else but a string or integer will return `FALSE` immediately.

Requirements

None besides functions from the standard C library which are always available.

Installation

Beginning with PHP 4.2.0 these function are enabled by default. For older versions you have to configure and compile PHP with `--enable-ctype`.

Runtime Configuration

This extension does not define any configuration directives.

Resource types

This extension does not define any resource types.

Predefined constants

This extension does not define any constants.

ctype_alnum (PHP 4 >= 4.0.4)

Check for alphanumeric character(s)

bool **ctype_alnum** (string text)

Returns `TRUE` if every character in *`text`* is either a letter or a digit, `FALSE` otherwise. In the standard `C` locale letters are just `[A-Za-z]`. The function is equivalent to `(ctype_alpha($text) || ctype_digit($text))`.

See also ctype_alpha(), ctype_digit(), and setlocale().

ctype_alpha (PHP 4 >= 4.0.4)

Check for alphabetic character(s)

bool **ctype_alpha** (string text)

Returns `TRUE` if every character in *`text`* is a letter from the current locale, `FALSE` otherwise. In the standard `C` locale letters are just `[A-Za-z]` and **ctype_alpha()** is equivalent to `(ctype_upper($text) || ctype_lower($text))`, but other languages have letters that are considered neither upper nor lower case.

See also ctype_upper(), ctype_lower(), and setlocale().

ctype_cntrl (PHP 4 >= 4.0.4)

Check for control character(s)

bool **ctype_cntrl** (string text)

Returns `TRUE` if every character in *`text`* has a special control function, `FALSE` otherwise. Control characters are e.g. line feed, tab, esc.

ctype_digit (PHP 4 >= 4.0.4)

Check for numeric character(s)

bool **ctype_digit** (string text)

Returns `TRUE` if every character in *`text`* is a decimal digit, `FALSE` otherwise.

See also ctype_alnum() and ctype_xdigit().

ctype_lower (PHP 4 >= 4.0.4)

Check for lowercase character(s)

bool **ctype_lower** (string text)

Returns TRUE if every character in *text* is a lowercase letter in the current locale.

See also ctype_alpha() and ctype_upper().

ctype_graph (PHP 4 >= 4.0.4)

Check for any printable character(s) except space

bool **ctype_graph** (string text)

Returns TRUE if every character in *text* is printable and actually creates visible output (no white space), FALSE otherwise.

See also ctype_alnum(), ctype_print(), and ctype_punct().

ctype_print (PHP 4 >= 4.0.4)

Check for printable character(s)

bool **ctype_print** (string text)

Returns TRUE if every character in *text* will actually create output (including blanks). Returns FALSE if *text* contains control characters or characters that do not have any output or control function at all.

See also ctype_cntrl(), ctype_graph(), and ctype_punct().

ctype_punct (PHP 4 >= 4.0.4)

Check for any printable character which is not whitespace or an alphanumeric character

bool **ctype_punct** (string text)

Returns TRUE if every character in *text* is printable, but neither letter, digit or blank, FALSE otherwise.

See also ctype_cntrl(), ctype_graph(), and **ctype_punct()**.

ctype_space (PHP 4 >= 4.0.4)

Check for whitespace character(s)

bool **ctype_space** (string text)

Returns TRUE if every character in *text* creates some sort of white space, FALSE otherwise. Besides the blank character this also includes tab, vertical tab, line feed, carriage return and form feed characters.

ctype_upper (PHP 4 >= 4.0.4)

Check for uppercase character(s)

bool **ctype_upper** (string text)

Returns `TRUE` if every character in *`text`* is a uppercase letter in the current locale.

See also ctype_alpha() and ctype_lower().

ctype_xdigit (PHP 4 >= 4.0.4)

Check for character(s) representing a hexadecimal digit

bool **ctype_xdigit** (string text)

Returns `TRUE` if every character in *`text`* is a hexadecimal 'digit', that is a decimal digit or a character from `[A-Fa-f]` , `FALSE` otherwise.

See also ctype_digit().

XVII. dba Datenbank (dbm-style) Abstraktions-Funktionen

Diese Funktionen bilden die Basis für den Zugriff auf Datenbanken vom Typ "Berkeley DB".

"dba" implementiert eine generelle Abstraktionsschicht für verschiedene dateibasierte Datenbanken. Der Funktionsumfang ist daher beschränkt auf eine gemeinsame Teilmenge des Funktionsumfangs aktueller Datenbanken (zum Beispiel Sleepycat Software's DB2 (http://www.sleepycat.com/), nicht zu verwechseln mit IBM's DB2 Software, die über die ODBC Funktionen unterstützt wird).

Das Verhalten der dba-Funktionen ist teilweise abhängig von der Implementation des zugrundeliegenden Datenbanksystems. Funktionen wie zum Beispiel dba_optimize() und dba_sync() funktionieren vielleicht bei einer Datenbank, und tun gar nichts bei einer Anderen.

Beim Aufruf von dba_open() oder dba_popen() muss einer der verfügbaren Handler-Namen aus der nachfolgenden Handler-Tabelle angegeben werden. Die tatsächlich verfügbaren Handler in Ihrer PHP-Installation können Sie mit der Funktion phpinfo() anzeigen lassen. (Um zum Zeitpunkt der PHP-Generierung die Unterstützung für einen der folgenden Handler verfügbar zu machen, fügen Sie bitte den jeweils angegebenen `--with-XXXX` Compilerschalter in ihren PHP-Konfigurations-Aufruf ein.)

Tabelle 1. Liste der DBA-Handler

Handler	Hinweise
`dbm`	Dbm ist der ursprüngliche und älteste Typ der "Berkeley DB" Datenbanken. Sie sollten diesen Typ wenn möglich vermeiden. Die in DB2 und gdbm enthaltenen dbm-Kompatibilitätsfunktionen werden ebenfalls nicht unterstützt, da sie nur auf Quellcode-Basis kompatibel sind, aber Dateien im originalen dbm-Format nicht bearbeiten können. (`--with-dbm`)
`ndbm`	Ndbm ist neuer und flexibler als Dbm. Es hat jedoch die meisten Einschränkungen von Dbm übernommen und ist daher ebenfalls nicht empfehlenswert. (`--with-ndbm`)
`gdbm`	Gdbm ist der GNU Datenbank Manager (ftp://ftp.gnu.org/pub/gnu/gdbm/). (`--with-gdbm`)
`db2`	DB2 ist Sleepycat Software's DB2 (http://www.sleepycat.com/). Es wird beschrieben als "Werkzeugsatz für eingebaute Unterstützung von Hochleistungs-Datenbanken für Standalone- und Client/Server- Applikationen". (`--with-db2`)
`db3`	DB3 ist Sleepycat Software's DB3 (http://www.sleepycat.com/). (`--with-db3`)

Handler	Hinweise
cdb	Cdb ist "ein schnelles, zuverlässiges und kompaktes Paket zum Erstellen und Auslesen konstanter Datenbanken." Es stammt vom Autor von qmail und kann hier (http://cr.yp.to/cdb.html) heruntergeladen werden. Da es für konstante Daten ausgelegt ist, werden nur Lese-Operationen unterstützt. (`--with-cdb`)

Beispiel 1. DBA Beispiel

```
<?php

$id = dba_open ("/tmp/test.db", "n", "db2");

if (!$id) {
    echo "dba_open schlug fehl\n";
    exit;
}

dba_replace ("key", "Dies ist ein Beispiel!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

DBA arbeitet binärsicher und besitzt keine willkürlichen Limits. Es erbt allerdings alle Einschränkungen von der zugrundeliegenden Datenbank-Implementation.

Alle dateibasierten Datenbanken müssen einen Weg zur Verfügung stellen, um den Dateimodus einer neu erstellten Datenbank zu bestimmen, soweit dies überhaupt möglich ist. Der Dateimodus wird gewöhnlich als viertes Argument an die Funktionen dba_open() oder dba_popen() übergeben.

Sie können auf alle Einträge in einer Datenbank sequentiell durch die Verwendung von dba_firstkey() und dba_nextkey() zugreifen. Während die Datenbank so durchlaufen wird, darf sie nicht verändert werden.

Beispiel 2. Durchlaufen einer Datenbank

```
<?php

# ...Oeffnen der Datenbank...

$key = dba_firstkey ($id);

while ($key != false) {
    if (...) { # den Schluessel fuer spaetere Aktionen merken
```

```
        $handle_later[] = $key;
    }
    $key = dba_nextkey ($id);
}

for ($i = 0; $i < count($handle_later); $i++)
    dba_delete ($handle_later[$i], $id);

?>
```

dba_close (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Datenbank schließen

void **dba_close** (int handle)

dba_close() schließt die offene Datenbank, die durch `handle` festgelegt wird, und gibt alle zugehörigen Ressourcen frei.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_close() besitzt keinen Rückgabewert.

Siehe auch dba_open() und dba_popen()

dba_delete (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Löschen des zu key gehörigen Eintrages

bool **dba_delete** (string key, int handle)

dba_delete() löscht den Eintrag mit dem Schlüsel `key` aus der durch `handle` spezifizierten Datenbank.

`key` ist der Schlüssel des Eintrags, der gelöscht wird.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_delete() hat als Rückgabewert `TRUE` oder `FALSE`, abhängig davon, ob der Löschvorgang erfolgreich war oder nicht.

Siehe auch dba_exists(), dba_fetch(), dba_insert(), und dba_replace().

dba_exists (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Überprüfen, ob Datensatz mit Schlüssel key existiert

bool **dba_exists** (string key, int handle)

dba_exists() überprüft, ob ein Datensatz mit dem Schlüssel `key` in der durch `handle` spezifizierten Datenbank existiert.

`Key` ist der Schlüssel, dessen Existenz überprüft werden soll.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_exists() hat als Rückgabewert `TRUE` oder `FALSE`, abhängig davon, ob der Schlüssel gefunden wurde oder nicht.

Siehe auch dba_fetch(), dba_delete(), dba_insert(), und dba_replace().

dba_fetch (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Datensatz für Schlüssel "key" auslesen

string **dba_fetch** (string key, int handle)

dba_fetch() liest die durch den Schlüssel `key` festgelegten Daten aus der durch `handle` spezifizierten Datenbank aus.

`Key` ist der Schlüssel, mit dem die Daten identifizert werden.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_fetch() hat als Rückgabewert entweder den zum Schlüssel `key` passenden Werte-String oder `FALSE`, abhängig davon, ob das Schlüssel/Wert-Paar gefunden wurde oder nicht.

Siehe auch dba_exists(), dba_delete(), dba_insert(), und dba_replace().

dba_firstkey (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Ersten Schlüssel bestimmen

string **dba_firstkey** (int handle)

dba_firstkey() gibt den ersten Schlüssel aus der durch `handle` spezifizierten Datenbank zurück und setzt den internen Schlüssel-Zeiger zurück. Dies erlaubt in Verbindung mit dba_nextkey() eine sequentielle Suche durch die gesamte Datenbank.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_firstkey() liefert als Rückgabewert den Schlüssel oder `FALSE`, abhängig davon, ob die Aktion erfolgreich war oder fehlschlug.

Siehe auch dba_nextkey() und Beispiel 2 im DBA-Überblick

dba_insert (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Datensatz einfügen

bool **dba_insert** (string key, string value, int handle)

dba_insert() fügt den Datensatz, der durch `key` und `value` beschrieben wird, in die durch `handle` spezifizierte Datenbank ein. Die Aktion schlägt fehl, wenn es bereits einen Eintrag mit dem gleichen `key` in der Datenbank gibt.

`key` ist der Schlüssel des Eintrags, der eingefügt werden soll.

`value` ist der Wert, der eingefügt werden soll.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_insert() hat als Rückgabewert `TRUE` oder `FALSE`, abhängig davon, ob die Aktion erfolgreich war oder nicht.

Siehe auch dba_exists() dba_delete() dba_fetch() dba_replace()

dba_nextkey (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Nachfolgenden Schlüssel bestimmen

string **dba_nextkey** (int handle)

dba_nextkey() gibt den nachfolgenden Schlüssel aus der durch *handle* spezifizierten Datenbank zurück und erhöht den internen Schlüssel-Zeiger.

handle ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_nextkey() hat als Rückgabewert den Schlüssel oder FALSE, abhängig davon, ob die Aktion erfolgreich war oder nicht.

Siehe auch dba_firstkey() und Beispiel 2 im DBA-Überblick

dba_popen (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Persistente Datenbank-Verbindung öffnen

int **dba_popen** (string path, string mode, string handler [, ...])

dba_popen() stellt eine persistente Datenbankinstanz für *path* mit *mode* her und benutzt dabei den Handler *handler*.

path ist ein gültiger Pfad in Ihrem Dateisystem.

Das *mode* Argument beschreibt den Modus, mit dem die Datenbank geöffnet werden soll. Es gibt 4 Möglichkeiten, eine DBA Datenbank zu öffnen:

"r" (read-only)

Eine existierende Datenbank soll nur lesend geöffnet werden.

"n" (new)

Eine neue Datenbank soll angelegt werden. Diese wird automatisch als read-write geöffnet. Eine existierende Datenbank wird inhaltlich gelöscht.

"c" (create)

Eine neue Datenbank soll angelegt werden. Diese wird automatisch als read-write geöffnet. Im Gegensatz zu "n" wird eine bereits existierende Datenbank inhaltlich nicht gelöscht, sondern nur für Lese- und Schreibzugriff geöffnet.

"w" (read-write)

Eine existierende Datenbank soll lesend und schreibend geöffnet werden.

handler ist der Name des Handlers, der für den Zugriff auf *path* benutzt werden soll. Die (oben durch *...* gekennzeichneten) optionalen Argumente der Funktion **dba_popen()** werden an ihn übergeben.

dba_popen() liefert als Rückgabewert eine positive Zugriffs-"handle" oder FALSE, abhängig davon, ob die Verbindung hergestellt werden konnte oder nicht.

Siehe auch dba_open() dba_close()

dba_open (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Verbindung zu einer Datenbank öffnen

int **dba_open** (string path, string mode, string handler [, ...])

dba_open() stellt eine Datenbankinstanz für `path` mit `mode` her und benutzt dabei `handler`.

`path` ist ein gültiger Pfad in Ihrem Dateisystem.

Das `mode` Argument beschreibt den Modus, mit dem die Datenbank geöffnet werden soll. Es gibt 4 Möglichkeiten, eine DBA Datenbank zu öffnen:

"r" (read-only)

Eine existierende Datenbank soll nur lesend geöffnet werden.

"n" (new)

Eine neue Datenbank soll angelegt werden. Diese wird automatisch als read-write geöffnet. Eine existierende Datenbank wird inhaltlich gelöscht.

"c" (create)

Eine neue Datenbank soll angelegt werden. Diese wird automatisch als read-write geöffnet. Im Gegensatz zu "n" wird eine bereits existierende Datenbank inhaltlich nicht gelöscht, sondern nur für Lese- und Schreibzugriff geöffnet.

"w" (read-write)

Eine existierende Datenbank soll lesend und schreibend geöffnet werden.

`handler` ist der Name des Handlers, der für den Zugriff auf `path` benutzt werden soll. Die (oben durch `...` gekennzeichneten) optionalen Argumente der Funktion **dba_open()** werden an ihn übergeben.

dba_open() liefert als Rückgabewert eine positive Zugriffs-"handle" oder `FALSE`, abhängig davon, ob die Verbindung hergestellt werden konnte oder nicht.

Siehe auch dba_popen() dba_close()

dba_optimize (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Optimieren einer Datenbank

bool **dba_optimize** (int handle)

dba_optimize() optimiert die zugrunde liegende Datenbank, die durch `handle` spezifiziert wird.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_optimize() hat als Rückgabewert `TRUE` oder `FALSE`, abhängig davon, ob die Optimierung erfolgreich war oder ob sie fehlschlug.

Siehe auch dba_sync()

dba_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Datensatz ersetzen oder einfügen

bool **dba_replace** (string key, string value, int handle)

dba_replace() ersetzt oder fügt in der durch `handle` spezifizierten Datenbank einen Datensatz ein, der durch den Schlüssel `key` und den Wert `value` beschrieben wird.

`key` ist der Schlüssel des Eintrags, der eingefügt werden soll.

`value` ist der Wert, der eingefügt werden soll.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_replace() liefert als Rückgabewert `TRUE` oder `FALSE`, abhängig davon, ob die Aktion erfolgreich war oder nicht.

Siehe auch dba_exists(), dba_delete(), dba_fetch(), und dba_insert().

dba_sync (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Datenbank synchronisieren

bool **dba_sync** (int handle)

dba_sync() synchronisiert die Datenbank, die durch `handle` spezifiziert wird. Dies kann möglicherweise einen physikalischen Schreibvorgang auf den Datenträger auslösen, falls dies unterstützt wird.

`handle` ist ein Datenbankhandle, das von dba_open() zurückgegeben wurde.

dba_sync() hat als Rückgabewert `TRUE` oder `FALSE`, abhängig davon, ob die Aktion erfolgreich war oder nicht.

Siehe auch dba_optimize()

XVIII. Datums- und Zeit-Funktionen

checkdate (PHP 3, PHP 4 >= 4.0.0)

Prüft eine gregorianische Datums- bzw. Zeitangabe auf Gültigkeit

int **checkdate** (int Monat, int Tag, int Jahr)

Gibt `TRUE` / wahr zurück, wenn das angegebene Datum gültig ist, sonst wird `FALSE` / falsch zurück gegeben. Prüft die Gültigkeit der angebenen Datumswerte. Ein Datum wird als gültig angesehen, wenn

- das Jahr zwischen 1 und 32767 (inclusive) ist,
- der Monat zwischen 1 und 12 (inclusive) ist und
- der Tag innerhalb der erlaubten Werte des angegebenen Monats ist. Schaltjahre werden berücksichtigt.

date (PHP 3, PHP 4 >= 4.0.0)

Formatiert ein(e) angegebene(s) Zeit/Datum

string **date** (string format [, int timestamp])

Gibt einen formatierten String anhand eines vorzugebenden Musters zurück. Dabei wird entweder der angegebene *`timestamp`* oder die gegenwärtige lokale Zeit berücksichtigt, wenn kein timestamp gegeben ist.

Die folgenden Platzhalter werden zur Formatierung benutzt:

- a - "am" oder "pm"
- A - "AM" oder "PM"
- B - Swatch-Internet-Zeit
- d - Tag des Monats, 2-stellig mit führender `NULL`: "01" bis "31"
- D - Tag der Woche als 3 Buchstaben, z.B. "Fri"
- F - Monat als ganzes Wort, z.B "January"
- g - Stunde im 12-Stunden-Format ohne führende `NULL`: "1" bis "12"
- G - Stunde im 24-Stunden-Format ohne führende `NULL`: "0" bis "23"
- h - Stunde im 12-Stunden-Format: "01" bis "12"
- H - Stunde im 24-Stunden-Format: "00" bis "23"
- i - Minuten: "00" bis "59"
- I (großes ’i’) - "1" bei Sommerzeit, sonst "0"
- j - Tag des Monats ohne führende `NULL`: "1" bis "31"
- l - (kleines ’L’) ausgeschriebener Tag der Woche, z.B. "Friday"
- L - boolscher Wert für Schaltjahre: "0" oder "1"
- m - Monat: "01" to "12"

- n - Monat ohne führende `NULL`: "1" bis "12"
- M - Monat als 3 Buchstaben, z.B. "Jan"
- O - Zeitdifferenz zur Greenwich Zeit in Stunden (z.B. +0200)
- r - Nach RFC 822 formattiertes Datum; z.B. "Sun, 7 Oct 2001 14:56:02 +0200" (seit PHP 4.0.4)
- s - Sekunden: "00" bis "59"
- S - Anhang der englischen Aufzählung, 2 Buchstaben, z.B. "th" [bei 4 für fourth], "nd" [bei 2 für second]
- t - Anzahl der Tage des angegebenen Monats: "28" bis "31"
- T - Zeitzoneneinstellung des Rechners, z.B. "MDT"
- U - Sekunden seit Beginn der UNIX-Epoche (1.1.1970)
- w - Wochentag beginnend mit "0" (Sonntag) bis "6" (Samstag)
- W - ISO-8601 Wochennummer des Jahres (Woche beginnt mit Montag) (seit PHP 4.1.0)
- Y - Jahr als vierstellige Zahl, z.B. "1999"
- y - Jahr als zweistellige Zahl, z.B. "99"
- z - Tag des Jahres: "0" bis "365"
- Z - Offset der Zeitzone gegen GTM ("-43200" bis "43200") in Minuten

Nicht definierte Zeichen werden unverändert zurück gegeben. Das "Z"-Format gibt "0" zurück beim Gebrauch von gmdate().

Beispiel 1. date() Beispiel

```
print (date("l dS of F Y h:i:s A"));
print ("July 1, 2000 is on a " . date ("l", mktime(0,0,0,7,1,2000)));
```

Es ist möglich, **date()** und mktime() gleichzeitig zu verwenden, um Datumsangaben der Zukunft/Vergangenheit herauszufinden.

Beispiel 2. date() und mktime() Beispiel

```
$morgen  = mktime(0, 0, 0, date ("m") , date ("d")+1, date("Y"));
$letztermonat = mktime(0, 0, 0, date("m")-1,date("d"), date("Y"));
$naechstesjahr = mktime(0, 0, 0, date("m"), date("d"), date("Y")+1);
```

Um Datumsangaben in anderen Sprach-/Zeitzonen-Formaten auszugeben, sollten sie die Funktionen setlocale() und strftime() nutzen.

Siehe auch gmdate() und mktime().

getdate (PHP 3, PHP 4 >= 4.0.0)

Gibt Datums- und Zeitinformationen zurück

array **getdate** (int timestamp)

Gibt ein assoziatives Array mit Datums- und Zeitangaben des angegebenen Timestamp zurück. Die Arrayelemente sind folgende:

- "seconds" - Sekunden
- "minutes" - Minuten
- "hours" - Stunden
- "mday" - Tag des Monats
- "wday" - numerischer Tag der Woche
- "mon" - Monat als Zahl
- "year" - Jahr als Zahl
- "yday" - Tag des Jahres als Zahlwert, z.B. "299"
- "weekday" - ausgeschriebener Wochentag, z.B. "Friday"
- "month" - ausgeschriebener Monatsname, z.B. "January"

gettimeofday (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt die aktuelle Zeit zurück

array **gettimeofday** (void)

Es handelt sich um eine Schnittstelle zum Systemaufruf von gettimeofday(2). Die Zeitangaben werden als assoziatives Array zurück gegeben. Folgende Arrayelemente sind vorhanden:

- "sec" - Sekunden
- "usec" - Microsekunden
- "minuteswest" - Minuten westlich von Greenwich
- "dsttime" - Art der Korrektur der Sommerzeit

gmdate (PHP 3, PHP 4 >= 4.0.0)

Formatiert eine GMT Zeitangabe

string **gmdate** (string format, int timestamp)

Entspricht der date() Funktion, ausser das als Zeitangabe immer Greenwich Mean Time (GMT) zurück gegeben wird. Steht ihr System in Deutschland (GMT + 01:00) wird im Beispiel unten (1. Zeile) "Jan 01 1998 00:00:00" ausgegeben, wogegen die 2. Zeile "Dec 31 1997 23:00:00" zurück gibt.

Beispiel 1. gmdate() Beispiel

```
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998))."\n";
echo gmdate("M d Y H:i:s", mktime (0,0,0,1,1,1998))."\n";
```

Siehe auch date(), mktime() und gmmktime().

gmmktime (PHP 3, PHP 4 >= 4.0.0)

Gibt den UNIX timestamp (Zeitstempel) als GMT zurück

int **gmmktime** (int Stunde, int Minute, int Secunde, int Monat, int Tag, int Jahr [, int is_dst])

Identisch zu mktime(). Die angegebenen Parameter werden aber als GMT-Datum/Zeit zurück gegeben.

gmstrftime (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Wandelt eine lokale Zeitangabe in GMT-Format um

string **gmstrftime** (string format, int timestamp)

Anmerkung: Erst ab Version 3.0.12

Bewirkt das gleiche wie strftime() mit dem Unterschied, dass die GMT-Zeit zurück gegeben wird. In Deutschland (GMT + 01:00) wird im folgenden Beispiel bei strftime "Dec 31 1998 00:00:00", bei gmstrftime dagegen "Dec 31 1998 23:00:00" ausgegeben.

Beispiel 1. gmstrftime() Beispiel

```
setlocale ('LC_TIME', 'ge');
echo strftime ("%b %d %Y %H:%M:%S", mktime(0,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S", mktime(0,0,0,12,31,98))."\n";
```

Siehe auch strftime().

localtime (PHP 4 >= 4.0.0)

Ermittelt die lokalen Zeitwerte

array **localtime** ([int timestamp [, bool is_associative]])

Die **localtime()**-Funktion gibt ein Array zurück, welches hinsichtlich seiner Struktur identisch ist mit dem des Funktionsaufrufs in C. Das erste (optionale) Argument von **localtime()** ist der UNIX-Zeitstempel (Timestamp). Fehlt dieser, wird die aktuelle Zeit genommen. Als zweites wird der Parameter `is_associative` übergeben. Ist dieser 0 oder nicht gesetzt, wird das Werte-Array als reguläres, numerisch indiziertes Array zurück gegeben. Ist er dagegen 1, liefert **localtime()** ein assoziatives Array zurück, welches alle Elemente der Struktur des localtime-Aufrufs in C enthält. Die Bezeichner der verschiedenen Schlüssel dieses assoziatien Arrays lauten:

- "tm_sec" - Sekunde
- "tm_min" - Minute
- "tm_hour" - Stunde
- "tm_mday" - Tag des Monats
- "tm_mon" - Monat des Jahres
- "tm_year" - Jahr (nicht Y2K sicher)
- "tm_wday" - Tag der Woche
- "tm_yday" - Tag des Jahres
- "tm_isdst" - für das Datum ist die Sommerzeit zu berücksichtigen

microtime (PHP 3, PHP 4 >= 4.0.0)

Gibt den aktuellen UNIX Timestamp/Zeitstempel in Mikrosekunden zurück

string **microtime** (void)

Gibt die Zeichenkette "msec sec" zurück, wobei sec die aktuelle Zeit ist, ausgedrückt als Anzahl der Sekunden seit Beginn der UNIX-Epoche (0:00:00 January 1, 1970 GMT). Msec stellt den Mikrosekunden-Teil dar. Diese Funktion ist nur auf Systemen möglich, die den Systemaufruf gettimeofday() unterstützen.

Siehe auch time().

mktime (PHP 3, PHP 4 >= 4.0.0)

Gibt den UNIX Timestamp/Zeitstempel für ein Datum zurück

int **mktime** (int Stunde, int Minute, int Sekunde, int Monat, int Tag, int Jahr [, int is_dst])

Warnung: Beachten sie die andersartige Anordnung der Argumente. Diese unterscheidet sich von der Reihenfolge des UNIX-Befehls mktime(). Das ist eine häufige Fehlerursache in Scripts.

Gibt den UNIX Timestamp/Zeitstempel an Hand der gegebenen Argumente zurück. Der Zeitstempel ist ein long-integer-Wert, der die Anzahl der Sekunden zwischen der UNIX Epoche (01.01.1970) und der angegebenen Zeit enthält.

Einzelne Argumente können von rechts nach links weggelassen werden. Sie werden dann ersetzt mit den Werten der lokalen Systemzeit bzw. des lokalen Systemdatums.

`is_dst` kann bei Sommerzeit auf 1 gesetzt werden. 0 bedeutet Winter-/Normalzeit, -1 (Vorgabewert) heisst, dass nicht bekannt ist, ob Sommer- oder Winterzeit herrscht.

Anmerkung: `is_dst` wurde in 3.0.10. eingeführt.

mktime() ist hilfreich bei Datumsberechnungen und -überprüfungen, z.B. wird versucht, automatisch das korrekte Datum zu berechnen. So wird in den folgenden Beispielen immer die Zeichenkette "Jan-01-1998" ausgegeben.

Beispiel 1. mktime() Beispiel

```
echo date ("M-d-Y", mktime(0,0,0,12,32,1997));
echo date ("M-d-Y", mktime(0,0,0,13,1,1997));
echo date ("M-d-Y", mktime(0,0,0,1,1,1998));
echo date ("M-d-Y", mktime(0,0,0,1,1,98));
```

`year` darf auch zweistellig angegeben werden. Werte von 0 bis 69 stehen dann für die Jahre 2000 bis 2069, und 70 bis 99 für die Jahre von 1970-1999 (auf Systemen, die time_t als 32Bit-Integer abbilden, liegt der gültige Bereich für `year` zwischen 1902 und 2037).

Der letzte Tag eines gegebenen Monats kann als der Tag "0" des nächsten Monats ausgedrückt werden (nicht als der Tag "-1"). Jedes der folgenden Beispiele gibt die Zeichenkette "Letzter Tag im Feb. 2000 ist der 29." zurück.

Beispiel 2. Letzter Tag des nächsten Monats

```
$lastday = mktime (0,0,0,3,0,2000);
echo strftime ("Letzter Tag im Feb. 2000 ist der %d.", $lastday);

$lastday=mktime(0,0,0,4,-31,2000);
echo strftime ("Letzter Tag im Feb. 2000 ist der %d.", $lastday);
```

Datumsangaben, bei denen das Jahr, der Monat und Tag gleich `NULL` sind, werden als ungültig angesehen (ansonsten als 30.11.1999, was nicht gewollt sein kann).

Siehe auch date() und time().

strftime (PHP 3, PHP 4 >= 4.0.0)

Formatiert eine Zeit-/Datumsangabe nach den lokalen Einstellungen

string **strftime** (string format [, int timestamp])

Gibt einen String zurück, der den angegebenen Formatierungs-Merkmalen entspricht. Dabei wird der gegebene Timestamp/Zeitstempel oder - falls dieser fehlt - die momentane lokale Zeit benutzt. Monats- und Wochentagsname wird entsprechend des per setlocale() eingestellten Wertes gesetzt.

Nachfolgend die im Formatierungsstring gültigen / bekannten Platzhalter:

- %a - abgekürzter Name des Wochentages, abhängig von der gesetzten Umgebung
- %A - ausgeschriebener Name des Wochentages, abhängig von der gesetzten Umgebung
- %b - abgekürzter Name des Monats, abhängig von der gesetzten Umgebung
- %B - ausgeschriebener Name des Monats, abhängig von der gesetzten Umgebung
- %c - Wiedergabewerte für Datum und Zeit, abhängig von der gesetzten Umgebung
- %C - Jahrhundert (Jahr geteilt durch 100, gekürzt auf Integer, Wertebereich 00 bis 99)
- %d - Tag des Monats als Zahl (Bereich 00 bis 31)
- %D - so wie %m/%d/%y
- %e - Tag des Monats als Dezimal-Wert, einstelligen Werten wird ein Leerzeichen voran gestellt (Wertebereich ´ 0´ bis ´31´)
- %h - so wie %b
- %H - Stunde als Zahl im 24-Stunden-Format (Bereich 00 bis 23)
- %I - Stunde als Zahl im 12-Stunden-Format (Bereich 01 bis 12)
- %j - Tag des Jahres als Zahl (Bereich 001 bis 366)
- %m - Monat als Zahl (Bereich 01 bis 12)
- %M - Minute als Dezimal-Wert
- %n - neue Zeile
- %p - entweder ‘am’ oder ‘pm’ (abhängig von der gesetzten Umgebung) oder die entsprechenden Zeichenketten der gesetzten Umgebung
- %r - Zeit in "am"- oder "pm"-Formatierung
- %R - Zeit in der 24-Stunden-Formatierung
- %S - Sekunden als Dezimal-Wert
- %t - Tabulator
- %T - aktuelle Zeit, genau wie %H:%M:%S
- %u - Tag der Woche als Dezimal-Wert [1,7], dabei ist 1 der Montag
- %U - Nummer der Woche des aktuellen Jahres als Dezimal-Wert, beginnend mit dem ersten Sonntag als erstem Tag der ersten Woche
- %V - Kalenderwoche (nach ISO 8601:1988) des aktuellen Jahres. Als Dezimal-Zahl mit dem Wertebereich 01 bis 53, wobei die Woche 01 die erste Woche mit mindestens 4 Tagen im aktuellen Jahr ist. Die Woche beginnt montags (nicht sonntags)
- %w - Wochentag als Dezimal-Wert, Sonntag ist 0
- %W - Nummer der Woche des aktuellen Jahres, beginnend mit dem ersten Montag als erstem Tag der ersten Woche
- %x - bevorzugte Datumswiedergabe (ohne Zeit), abhängig von der gesetzten Umgebung
- %X - bevorzugte Zeitwiedergabe (ohne Datum), abhängig von der gesetzten Umgebung
- %y - Jahr als 2-stellige-Zahl (Bereich 00 bis 99)
- %Y - Jahr als 4-stellige-Zahl
- %Z - Zeitzone
- %% - ein %-Zeichen

Beispiel 1. strftime() Beispiel

```
setlocale ("LC_TIME", "C");
print (strftime ("%A in Finnland ist "));
setlocale ("LC_TIME", "fi_FI");
print (strftime ("%A, in Frankreich "));
setlocale ("LC_TIME", "fr_CA");
print (strftime ("%A und in Deutschland "));
setlocale ("LC_TIME", "de_DE");
print (strftime ("%A.\n"));
```

Diese Beispiel funktioniert, wenn sie die jeweilige Umgebung in ihrem System eingestellt haben.

Siehe auch setlocale() und mktime().

time (PHP 3, PHP 4 >= 4.0.0)

Gibt den gegenwärtigen UNIX-Timestamp/Zeitstempel zurück

int **time** (void)

Gibt die seit Beginn der UNIX-Epoche (Januar 1 1970 00:00:00 GMT) bis jetzt vergangenen Sekunden zurück.

Siehe auch date().

strtotime (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Wandelt ein beliebiges Datum (englisches Format) in einen Unix-Zeitstempel (timestamp) um

int **strtotime** (string time [, int now])

Diese Funktion erwartet einen String mit einem Datum im englischen Format und versucht, dieses in einen UNIX-Timestamp umzusetzen.

Beispiel 1. strtotime()-Beispiele:

```
echo strtotime ("now") . "\n";
echo strtotime ("10 September 2000") . "\n";
echo strtotime ("+1 day") . "\n";
echo strtotime ("+1 week") . "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds") . "\n";
```

XIX. dBase Funktionen

Diese Funktionen erlauben ihnen den Zugriff auf Datensätze, die im dBase-Format (also in dBase-Datenbanken) (dbf) vorliegen.

Es gibt keine Unterstützug für Indizes oder Memo-Felder. Auch Sperrfunktionen für Datensätze / -banken sind nicht vorhanden. Zwei gleichzeitig ablaufende Webserver-Prozesse, die die selbe dBase-Datei bearbeiten, werden ihre Datenbank zerstören.

Anders als bei SQL-Datenbanken können in dBase-"Datenbanken" die Felddefinitionen nicht nachträglich geändert werden. Nachdem die dbf-Datei erzeugt wurde, sind die Definitionen festgelegt. Es werden keine Index-Funktionen unterstützt, die Abfragen beschleunigen oder etwa ihre Datenorganisation unterstützen. dBase-Dateien sind einfache sequentielle Dateien mit fester Datensatzlänge. Neue Datensätze werden am Dateiende angehängt und gelöschte Datensätze werden beibehalten, bis sie dbase_pack() aufrufen.

Wir empfehlen, keine Produktions-DBF-Datenbanken zu benutzen. Wählen sie zur Nutzung von Produktions-Daten mit PHP nur echte SQL-Server-Datenbanken; MySQL oder Postgres sind dabei eine gute Wahl. dBase-Unterstützung durch PHP dient hauptsächlich dem Im- und Export von Daten zu oder aus ihrer Web-Datenbank, wobei das DBF-Datei-Format vor allem wegen der Windows-Tabellenkalulationen und -Organizer unterstützt wird. Im- und Export ist der einzige Grund für die dBase-Unterstützung durch PHP.

dbase_create (PHP 3, PHP 4 >= 4.0.0)

Erzeugt eine dBase-Datenbank

int **dbase_create** (string filename, array fields)

Der `fields`-Parameter ist ein Array, das aus weiteren Arrays besteht. Jedes Array beschreibt das Format eines Feldes der Datenbank. Jedes Feld-Array besteht aus einem Namen, einem Zeichen zur Festlegung des Feldtyps, der Länge und dem Wert für die Anzahl der Nachkommastellen.

Die verfügbaren Feldtypen sind:

L

Boolean. Diese haben keine Länge oder Nachkommstellen.

M

Memo. (Beachten sie, dass diese von PHP nicht unterstützt werden.) Auch diese haben keine Länge oder Nachkommastellen.

D

Datum (gespeichert als JJJJMMTT). Ebenfalls keine Länge und Nachkommastellen.

N

Zahl. Diese haben sowohl eine Angabe der Länge als auch der Nachkommastellen.

C

Zeichenkette / String / Charakter. Hier ist eine Längenangabe erforderlich.

Konnte die Datenbank erfolgreich erzeugt werden, wird ein dbase_identifier zurück gegeben, anderenfalls `FALSE` / falsch.

Beispiel 1. Erzeugen einer dBase-Datenbank-Datei

```
// "Datenbank"-Name
$dbname = "/tmp/test.dbf";

// Datenbank-"Definitionen"
$def =
    array(
      array("Datum",       "D"),
      array("Name",        "C",  50),
      array("Alter",       "N",   3, 0),
      array("Email",       "C", 128),
      array("IstMitglied", "L")
    );

// Erzeugung
if (!dbase_create($dbname, $def))
  print "<strong>Error!</strong>";
```

dbase_open (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine dBase-Datenbank

int **dbase_open** (string filename, int flags)

Die Flags entsprechen denen des open()-Systemaufrufs (normalerweise bedeutet 0 Nur-Lesen, 1 Nur-Schreiben und 2 Lesen und Schreiben).

Gibt einen dbase_identifier für die geöffnete Datenbank zurück oder `FALSE` / falsch, wenn die Datenbank nicht geöffnet werden konnte.

dbase_close (PHP 3, PHP 4 >= 4.0.0)

Schließt eine dBase-Datenbank

bool **dbase_close** (int dbase_identifier)

Schließt die durch *`dbase_identifier`* angegebene Datenbank.

dbase_pack (PHP 3, PHP 4 >= 4.0.0)

Entfernt die als gelöscht markierten Datensätze aus der Datenbank

bool **dbase_pack** (int dbase_identifier)

Entfernt die in der angegebenen Datenbank als gelöscht markierten Datensätze endgültig (physisch). Betroffen sind alle vorher per dbase_delete_record() als gelöscht markierten Datensätze.

dbase_add_record (PHP 3, PHP 4 >= 4.0.0)

Hängt einen neuen Datensatz an eine dBase-Datenbank an

bool **dbase_add_record** (int dbase_identifier, array record)

Hängt den per *`record`* definierten Datensatz an die Datenbank an. Wenn die Anzahl der Einträge im angegebenen Record nicht gleich der Anzahl der Felder in der Datenbank ist, wird dieser Befehl scheitern und `FALSE` / falsch zurück geben.

dbase_replace_record (PHP 3>= 3.0.11, PHP 4 >= 4.0.0)

Ersetzt einen Datensatz in einer dBase-Datenbank

bool **dbase_replace_record** (int dbase_identifier, array record, int dbase_record_number)

Überschreibt die Daten des Datensatzes *`record_number`* mit den Daten aus *`record`* in der Datenbank. Falls die Anzahl der definierten Einträge ungleich der Anzahl der Datenfelder in dem

angesprochenen Datensatz ist, wird die Ausführung dieses Befehls abgewiesen und `FALSE` / falsch zurück gegeben.

`dbase_record_number` ist ein Integer-Wert, der von 1 bis zur Anzahl der Datensatz-Anzahl der Datenbank reicht (wie er bei Aufruf der Funktion dbase_numrecords() zurück gegeben wird).

dbase_delete_record (PHP 3, PHP 4 >= 4.0.0)

Markiert einen Datensatz als gelöscht

bool **dbase_delete_record** (int dbase_identifier, int record)

Markiert den Datensatz *`record`* in der Datenbank als gelöscht. Um ihn wirklich (physisch) zu entfernen, müssen sie noch die Funktion dbase_pack() aufrufen.

dbase_get_record (PHP 3, PHP 4 >= 4.0.0)

Liest einen Datensatz aus einer dBase-Datenbank

array **dbase_get_record** (int dbase_identifier, int record)

Gibt die Daten von *`record`* in einem Array zurück. Das Array hat einen bei 0 beginnenden Index und enthält einen zusätzlichen Bereich der mit 'deleted' bezeichnet wird. Dieser ist auf 1 gesetzt, wenn der Datensatz als gelöscht markiert ist (siehe dbase_delete_record()).

Jedes Feld wird in den entsprechenden PHP-Typ umgesetzt (Datumsangaben werden zu Zeichenketten / Strings).

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Gibt einen Datensatz einer dBase-Datenbank als assoziatives Array zurück

array **dbase_get_record_with_names** (int dbase_identifier, int record)

Gibt die Daten von *`record`* in einem assoziativen Array zurück. Das Array enthält auch einen assoziativen Bereich, der 'deleted' heisst und auf 1 steht, wenn der Datensatz als gelöscht markiert ist (siehe dbase_delete_record()).

Jedes Feld wird in den entsprechenden PHP-Typ konvertiert (Datumsangaben werden zu Zeichenketten / Strings).

dbase_numfields (PHP 3, PHP 4 >= 4.0.0)

Stellt fest, wieviele Felder eine dBase-Datenbank hat

int **dbase_numfields** (int dbase_identifier)

Gibt die Anzahl der Datenfelder (Spalten) in der angegebenen Datenbank zurück. Feld-Nummern sind Werte zwischen 0 und dbase_numfields($db)-1, wogegen die Datensatz-Werte zwischen 1 und dbase_numrecords($db) sein können.

Beispiel 1. Gebrauch von dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
  print $rec[$i]."<br>\n";
}
```

dbase_numrecords (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Anzahl der Datensätze einer dBase-Datenbank

int **dbase_numrecords** (int dbase_identifier)

Gibt die Anzahl der Datensätze (Zeilen) in der angegebenen Datenbank zurück. Datensatz-Nummern können Werte zwischen 1 und dbase_numrecords($db) sein, wogegen Feld-Werte zwischen 0 und dbase_numfields($db)-1 sein können.

XX. DBM Datenbankfunktionen

Diese Funktionen erlauben Ihnen den Zugriff auf Datensätze die im DBM-Format vorliegen. Im Gegensatz zu den relationalen Datenbanken (mySQL, etc.) werden die Datensätze in Form von ISAM-Sätzen gespeichert. DBM Datenbanken werden von verschiedenen Systemfunktionen als interne Arbeitsbibliothek benützt. Diese Datenbank-Art wird auch von der Berkeley DB und GDBM verwendet.

Beispiel 1. Beispiel für ein DBM Script

```
$dbm = dbmopen ("lastseen", "w");
if (dbmexists ($dbm, $userid)) {
    $last_seen = dbmfetch ($dbm, $userid);
} else {
    dbminsert ($dbm, $userid, time());
}
do_stuff();
dbmreplace ($dbm, $userid, time());
dbmclose ($dbm);
```

dbmopen (PHP 3, PHP 4 >= 4.0.0)

Öffnen einer DBM Datenbank

int **dbmopen** (string filename, string flags)

Das erste Argument ist der vollqualifizierte Dateiname der DBM Datenbank, die geöffnet werden soll. Das zweite Argument ist der Modus, mit dem die DBM Datenbank geöffnet werden soll. Es gibt 4 Möglichkeiten, eine DBM Datenbank zu öffnen:

"r" (read-only)

Eine existierende Datenbank soll nur lesend geöffnet werden.

"n" (new)

Eine neue Datenbank soll angelegt werden. Diese wird automatisch als read-write geöffnet. Eine existierende Datenbank wird inhaltlich gelöscht.

"c" (create)

Eine neue Datenbank soll angelegt werden. Diese wird automatisch als read-write geöffnet. Im Gegensatz zu "n" wird eine bereits existierende Datenbank inhaltlich nicht gelöscht, sondern nur "read-writeable" geöffnet.

"w" (read-write)

Eine existierende Datenbank soll lesend und schreibend geöffnet werden.

Wenn die angegebene DBM Datenbank geöffnet werden konnte, wird ein Identifier zurückgegeben. Dieser Identifier muss für alle weiteren Aufrufe, die an die DBM Datenbank getätigt werden, mitgegeben werden. Wenn das Öffenen der DBM Datenbank fehlgeschlagen ist, wird das Ergebnis `FALSE` zurückgegeben.

Wenn die NDBM Unterstützung aktiviert wird, werden jeweils zwei Dateien `filename.dir` und `filename.pag` erzeugt. GDBM, ebenso wie der interne "Flat-File" Mechanismus und die Berkeley DB, verwenden nur eine Datei (und diese heisst bei der Berkeley DB `filename.db`). Bitte beachten Sie, dass PHP seine Dateisperren zusätzlich zu den vorhanden DBM-Bibliothekseigenen Routinen verwendet. PHP löscht auch die `.lck` Dateien nicht wieder. Es verwendet diese Datei nur als zusätzliches Mittel für Dateilocking. Wenn Sie mehr über DBM-Datenbanken wissen wollen, sehen Sie in den UNIX-Manualseiten nach, oder schauen bei GNU's GDBM (ftp://ftp.gnu.org/pub/gnu/gdbm/) nach.

dbmclose (PHP 3, PHP 4 >= 4.0.0)

Schließen einer DBM Datenbank

bool **dbmclose** (int dbm_identifier)

Gibt Zugiffssperren auf Datensätze frei und schließt die angegebene DBM Datenbank.

Liefert `TRUE` bei Erfolg bzw. `FALSE` bei Fehler.

dbmexists (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob ein Eintrag für den angegebenen Schlüssel in der DBM Datenbank vorhanden ist.

bool **dbmexists** (int dbm_identifier, string key)

Wenn in der DBM Datenbank ein Eintrag mit dem angegebenen Schlüssel *key* vorhanden ist, wird als Ergebnis `TRUE` zurückgegeben.

dbmfetch (PHP 3, PHP 4 >= 4.0.0)

Holt einen Datensatz mit dem angegebenen Schlüssel aus der DBM Datenbank.

string **dbmfetch** (int dbm_identifier, string key)

Liefert den Datensatz, der dem *key* in der DBM Datenbank entspricht.

dbminsert (PHP 3, PHP 4 >= 4.0.0)

Schreibt einen Datensatz mit dem angegebenen Schlüssel in die DBM Datenbank.

int **dbminsert** (int dbm_identifier, string key, string value)

Schreibt den Datensatz mit angegebenen Schlüssel in die Datenbank.

Ergebniswerte:

-1

nur Leseberechtigung.

0

Aktion erfolgreich beendet.

1

Ein Eintrag mit dem angegebenen Schlüssel existiert bereits in der Datenbank.

dbmreplace (PHP 3, PHP 4 >= 4.0.0)

Ersetzt einen Datensatz mit dem angegebenen Schlüssel in der DBM Datenbank.

bool **dbmreplace** (int dbm_identifier, string key, string value)

Schreibt einen Datensatz mit dem angegebenen Satzschlüssel in die Datenbank bzw. ersetzt ihn.

Wenn ein Eintrag mit den angegebenen Schlüssel vorhanden ist, wird dieser überschrieben. Ansonsten wird ein Datensatz mit mit den Schlüssel neu angelegt.

dbmdelete (PHP 3, PHP 4 >= 4.0.0)

Löscht einen Datensatz mit dem angegebenen Schlüssel aus der DBM Datenbank.

bool **dbmdelete** (int dbm_identifier, string key)

Löscht einen vorhanden Datensatz, der dem `key` entspricht, aus der Datenbank.

Wenn kein Datensatz zum Löschen gefunden wurde, wird der Returncode `FALSE` zurückgegeben.

dbmfirstkey (PHP 3, PHP 4 >= 4.0.0)

Gibt den ersten Schlüssel der DBM Datenbank zurück.

string **dbmfirstkey** (int dbm_identifier)

Gibt den ersten Schlüssel der DBM Datenbank zurück.

Anmerkung: Achtung: Es gibt keine Garantie für eine bestimmte Reihenfolge innerhalb der Datenbank, da die Datenbank auf einen Hash-Algorithmus aufgebaut ist.

dbmnextkey (PHP 3, PHP 4 >= 4.0.0)

Stellt den nächsten Schlüssel aus den Datenbank zur Verfügung.

string **dbmnextkey** (int dbm_identifier, string key)

Gibt den auf `key` folgenden Schlüssel an den Aufrufer zurück. Um alle Schlüssel einer DBM Datenbank auszulesen, wird dbmfirstkey(), gefolgt von wiederholten Aufrufen von **dbmnextkey()** verwendet, wie im folgenden Script:

Beispiel 1. Ausgeben aller Schlüssel mit Datensätzen einer DBM Datenbank

```
$key = dbmfirstkey ($dbm_id);
while ($key) {
    echo "$key = " . dbmfetch ($dbm_id, $key) . "\n";
    $key = dbmnextkey ($dbm_id, $key);
}
```

dblist (PHP 3, PHP 4 >= 4.0.0)

Liefert eine Beschreibung, welches DBM-Format verwendet wird.

string **dblist** (void)

XXI. dbx Funktionen

Das dbx Modul ist eine Datenbank-Abstraktionsschicht (db 'X', wobei 'X' eine unterstützte Datenbank ist). Die dbx Funktionen ermöglichen Ihnen den Zugriff auf alle unterstützten Datenbanken mit einer einzigen Aufrufkonvention. Diese Funktionen sind nur verfügbar, wenn PHP mit der Option `--enable-dbx`, sowie allen Optionen für die zu benutzenden Datenbanken (z.B. für MySQL `--with-mysql`) konfiguriert und kompiliert wurde. Die dbx Funktionen stellen selbst keinen direkten Kontakt zu den Datenbanken her, sondern benutzen dafür jene Module, welche die jeweilige Datenbank unterstützen. Um eine Datenbank mit dem dbx Modul benutzen zu können, muss das Modul in PHP gelinkt oder geladen sein, und das Datenbankmodul muss vom dbx-Modul unterstützt werden. Derzeit werden MySQL, PostgreSQL, Microsoft SQL Server, FrontBase, Sybase-CT und ODBC unterstützt, und weitere werden (hoffentlich bald :-) folgen.

Eine Dokumentation, wie dbx weitere Datenbankunterstützung hinzugefügt werden kann, finden Sie unter http://www.guidance.nl/php/dbx/doc/.

dbx_close (PHP 4 >= 4.0.6)

Schließt eine offene Verbindung/Datenbank

bool **dbx_close** (object link_identifier)

Gibt bei Erfolg `TRUE`, oder im Fehlerfall `FALSE` zurück.

Beispiel 1. dbx_close()

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Fehler beim Verbinden");

print("Verbindung hergestellt");
dbx_close($link);
?>
```

Anmerkung: Konsultieren Sie bitte auch die modulspezifische Dokumentation.

Siehe auch dbx_connect().

dbx_connect (PHP 4 >= 4.0.6)

Öffnet eine Verbindung/Datenbank

object **dbx_connect** (mixed module, string host, string database, string username, string password [, int persistent])

dbx_connect() gibt bei Erfolg ein Objekt, oder im Fehlerfall `FALSE` zurück. Konnte eine Verbindung hergestellt, jedoch keine Datenbank ausgewählt werden, wird die Verbindung wieder geschlossen, und `FALSE` zurückgegeben. Wird der Parameter *`persistent`* auf `DBX_PERSISTENT` gesetzt, wird eine persistente Verbindung hergestellt.

Der Parameter *`module`* kann entweder einen String, oder eine Konstante enthalten, auch wenn letzteres vorgezogen wird. Die möglichen Werte sind nachstehend aufgelistet, aber bedenken Sie bitte, dass diese nur funktionieren, wenn das Modul gerade geladen ist.

- `DBX_MYSQL` oder "mysql"
- `DBX_ODBC` oder "odbc"
- `DBX_PGSQL` oder "pgsql"
- `DBX_MSSQL` oder "mssql"
- `DBX_FBSQL` oder "fbsql" (seit PHP 4.1.0 verfügbar)
- `DBX_SYBASECT` oder "sybase_ct" (nur CVS)

Die Parameter *host*, *database*, *username* und *password* werden erwartet, werden aber abhängig von den Verbindungsfunktionen für das abstrahierte Modul nicht immer verwendet.

Das zurückgegebene `Objekt` hat drei Eigenschaften:

database

Das ist der Name der derzeit ausgewählten Datenbank.

handle

Das ist ein gültiges Handle für die verbundene Datenbank, und als solche kann es wenn nötig auch in modulspezifischen Funktionen verwendet werden.

```
$link = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) wäre hier besser
```

module

Dies ist die oben erwähnte Modulnummer, und wird nur dbx-intern verwendet.

Beispiel 1. dbx_connect()

```
<?php
$link = dbx_connect (DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die ("Fehler beim Verbinden");

print ("Verbindung hergestellt");
dbx_close ($link);
?>
```

Anmerkung: Konsultieren Sie bitte auch die modulspezifische Dokumentation.

Siehe auch dbx_close().

dbx_error (PHP 4 >= 4.0.6)

Liefert die Fehlermeldung des letzten Funktionsaufrufes in dem Modul (nicht nur in der Verbindung).

string **dbx_error** (object link_identifier)

dbx_error() gibt eine Zeichenkette mit der Fehlermeldung des letzten Funktionsaufrufes von dem abstrahierten Modul (z.B. MySQL-Modul) zurück. Bestehen mehrere Verbindungen zu dem gleichen Modul, wird nur der letzte Fehler zurückgegeben. Bestehen Verbindungen zu verschiedenen Modulen, wird der letzte Fehler von dem im Parameter *link_identifier* angegebenen Modul zurückgegeben.

Beispiel 1. dbx_error()

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Fehler beim Verbinden");

$result = dbx_query($link, "select id from non_existing_table");
if ( $result == 0 ) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

Anmerkung: Konsultieren Sie bitte auch die modulspezifische Dokumentation.

Die Fehlermeldung des Microsoft SQL Server ist übrigens das Ergebnis der Funktion mssql_get_last_message().

dbx_query (PHP 4 >= 4.0.6)

Sendet eine Abfrage und holt alle Ergebnisse (falls vorhanden)

object **dbx_query** (object link_identifier, string sql_statement [, long flags])

dbx_query() gibt bei Erfolg ein Objekt oder `1` zurück, oder `0` im Fehlerfall. Das Objekt wird nur zurückgegeben, wenn die in *sql_statement* angegebene Abfrage eine Ergebnisliste liefert.

Beispiel 1. Wie mit dem gelieferten Wert umgegangen wird

```
<?php
$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Fehler beim Verbinden");

$result = dbx_query($link, 'SELECT id, parentid, description FROM table');

if ( is_object($result) ) {
    // ... tue hier irgendetwas, detaillierte Beispiele siehe unten ...
    // erst die Ausgabe der Feldnamen und Typen
    // dann die Ausgabe einer Tabelle mit den gelieferten Werten
}
else if ( $result == 1 ) {
    echo("Abfrage wurde erfolgreich ausgeführt, jedoch keine Ergebnisse");
```

```
}
else {
    exit("Fehler bei Abfrage");
}

dbx_close($link);
?>
```

Der Parameter *`flags`* wird verwendet, um die Menge der zu liefernden Informationen zu steuern. Die folgenden Konstanten können mit dem Bit-Operator (|) beliebig kombiniert werden:

`DBX_RESULT_INDEX`

Dieser Flag ist *immer* gesetzt, d.h. dass zurückgegebene Objekt hat eine Eigenschaft data, welche aus einem zweidimensionalen, numerisch indizierten Array besteht. Zum Beispiel steht in `data[2][3]` die `2` für die Reihen- (bzw. Datensatz-) Nummer, und `3` steht für die Spalten- (bzw. Feld- Nummer). Die erste Reihe und Spalte haben den Index 0.

Ist `DBX_RESULT_ASSOC` ebenfalls angegeben, so enthält das zurückgegebene Objekt auch Informationen im Zusammenhang mit `DBX_RESULT_INFO`, selbst wenn es nicht angegeben wurde.

`DBX_RESULT_INFO`

Liefert Informationen über Spalten, wie Feldnamen und Feldtypen.

`DBX_RESULT_ASSOC`

Dies bewirkt, dass in der data Eigenschaft des zurückgegebenen Objektes auf die Werte eines Feldes mit den entsprechenden Spaltennamen als Schlüssel zugegriffen werden kann.

Assoziierte Ergebnisse sind eigentlich Referenzen zu den numerisch indizierten Daten, weshalb eine Änderung von `data[0][0]` auch den Inhalt von `data[0]['feldname_für_erste_spalte']` betrifft.

Beachten Sie, dass `DBX_RESULT_INDEX` unabhängig von dem aktuell verwendeten Wert des Parameters *`flags`* immer verwendet wird. Das heißt, dass es effektiv nur die folgenden Kombinationen gibt:

- `DBX_RESULT_INDEX`
- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO`
- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO` | `DBX_RESULT_ASSOC` - dies ist Standard, wenn *`flags`* nicht angegeben ist.

Das zurückgegebene Objekt hat abhängig von *`flags`* vier oder fünf Eigenschaften:

handle

Das ein gültiges Handle für die verbundene Datenbank, und als solche kann es (wenn nötig) auch in modulspezifischen Funktionen verwendet werden.

```
$result = dbx_query ($link, "SELECT id FROM table");
mysql_field_len ($result->handle, 0);
```

cols und rows

Diese enthalten die Anzahl der Spalten (oder Felder) bzw. Reihen (oder Datensätze).

```
$result = dbx_query ($link, 'SELECT id FROM table');
echo $result->rows; // Anzahl der Datensätze
echo $result->cols; // Anzahl der Felder
```

info (optional)

Dies wird nur zurückgegeben, wenn im Parameter *`flags`* entweder `DBX_RESULT_INFO` oder `DBX_RESULT_ASSOC` spezifiziert sind. Es ist ein zweidimensionales Array mit zwei Reihen (`name` und `type`), welches die Spalteninformationen enthält.

Beispiel 2. Auflistung von Name und Typ jedes Feldes

```
$result = dbx_query ($link, 'SELECT id FROM table',
                     DBX_RESULT_INDEX | DBX_RESULT_INFO);

for ($i = 0; $i < $result->cols; $i++ ) {
    echo $result->info['name'][$i] . "\n";
    echo $result->info['type'][$i] . "\n";
}
```

data

Diese Eigenschaft enthält die aktuellen Ergebnisdaten, möglicherweise auch mit den Spaltennamen assoziiert, was jedoch vom Parameter *`flags`* abhängig ist. Wenn `DBX_RESULT_ASSOC` gesetzt ist, können Sie `$result->data[2]["feldname"]` verwenden.

Beispiel 3. Ausgabe des Inhaltes der data Eigenschaft in eine HTML Tabelle

```
$result = dbx_query ($link, 'SELECT id, parentid, description FROM table');

echo "<table>\n";
foreach ( $result->data as $row ) {
    echo "<tr>\n";
    foreach ( $row as $field ) {
```

```
            echo "<td>$field</td>";
        }
        echo "</tr>\n";
    }
    echo "</table>\n";
```

Anmerkung: Konsultieren Sie bitte auch die modulspezifische Dokumentation.

Siehe auch dbx_connect().

dbx_sort (PHP 4 >= 4.0.6)

Sortiert das Ergebnis einer dbx-Abfrage mittels einer Benutzerfunktion

bool **dbx_sort** (object result, string user_compare_function)

Gibt bei Erfolg `TRUE`, oder im Fehlerfall `FALSE` zurück.

Anmerkung: Wenn möglich ist es besser, die `ORDER BY SQL` Klausel anstatt von **dbx_sort()** zu verwenden.

Beispiel 1. dbx_sort()

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Fehler beim Verbinden");

$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
    // Die Daten in $result sind nun nach id geordnet

dbx_sort ($result, "user_re_order");
    // Die Daten in $result sind nun nach parentid (absteigend) geordnet,
    // dann nach id

dbx_close ($link);
?>
```

Siehe auch dbx_compare().

dbx_compare (PHP 4 >= 4.1.0)

Vergleicht zwei Reihen für Sortierzwecke

int **dbx_compare** (array row_a, array row_b, string column_key [, int flags])

dbx_compare() gibt 0 zurück, wenn `row_a[$column_key]` gleich `row_b[$column_key]` ist. Ist der erste Wert größer oder kleiner als der zweite, wird `1` bzw. `-1` zurückgegeben (oder umgekehrt, wenn der Parameter *`flag`* auf `DBX_CMP_DESC` gesetzt ist). **dbx_compare()** ist eine Hilfsfunktion für dbx_sort(), um das Erstellen bzw. Verwenden der Benutzerfunktion zu vereinfachen.

Der Parameter *`flags`* spezifiziert die Vergleichsrichtung:

- `DBX_CMP_ASC` - aufsteigende Reihenfolge
- `DBX_CMP_DESC` - absteigende Reihenfolge

und den bevorzugten Vergleichstyp:

- `DBX_CMP_NATIVE` - keine Typenkonvertierung
- `DBX_CMP_TEXT` - vergleicht die Werte als Strings
- `DBX_CMP_NUMBER` - vergleicht die Werte numerisch

Eine der Vergleichs-, und eine der Typkonstanten können mit dem Bit-Operator OR (|) kombiniert werden. Die Standardwerte für *`flags`* sind `DBX_CMP_ASC` | `DBX_CMP_NATIVE`.

Beispiel 1. dbx_compare()

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Fehler beim Verbinden");

$result = dbx_query ($link, "SELECT id, parentid, description FROM table OR-
DER BY id");
    // die Daten in $result sind nun nach id gereiht

dbx_sort ($result, "user_re_order");
    // die Daten in $result sind nun nach parentid (absteigend) gereiht,
    // und dann nach id
```

```
dbx_close ($link);
?>
```

Siehe auch dbx_sort().

XXII. DB++ Functions

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

db++, made by the german company Concept asa (http://www.concept-asa.de/), is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Requirements

This extension relies on external client libraries so you have to have a db++ client installed on the system you want to use this extension on.

Concept asa (http://www.concept-asa.de/) provides db++ Demo versions (http://www.concept-asa.de/down-eng.html) and documentation (http://www.concept-asa.de/downloads/doc-eng.tar.gz) for Linux, some other UNIX versions. There is also a Windows version of db++, but this extension doesn't support it (yet).

Installation

In order to build this extension yourself you need the db++ client libraries and header files to be installed on your system (these are included in the db++ installation archives by default). You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default paths `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this:

`--with-dbplus=/your/installation/path`.

Runtime Configuration

This extension does not define any configuration directives.

Resource Types

dbplus_relation

Most db++ functions operate on or return *`dbplus_relation`* resources. A *`dbplus_relation`* is a handle to a stored relation or a relation generated as the result of a query.

Predefined Constants

db++ error codes

Tabelle 1. DB++ Error Codes

PHP Constant	db++ constant	meaning
DBPLUS_ERR_NOERR	ERR_NOERR	Null error condition
DBPLUS_ERR_DUPLICATE	ERR_DUPLICATE	Tried to insert a duplicate tuple
DBPLUS_ERR_EOSCAN	ERR_EOSCAN	End of scan from rget()
DBPLUS_ERR_EMPTY	ERR_EMPTY	Relation is empty (server)
DBPLUS_ERR_CLOSE	ERR_CLOSE	The server can't close
DBPLUS_ERR_WLOCKED	ERR_WLOCKED	The record is write locked
DBPLUS_ERR_LOCKED	ERR_LOCKED	Relation was already locked
DBPLUS_ERR_NOLOCK	ERR_NOLOCK	Relation cannot be locked
DBPLUS_ERR_READ	ERR_READ	Read error on relation
DBPLUS_ERR_WRITE	ERR_WRITE	Write error on relation
DBPLUS_ERR_CREATE	ERR_CREATE	Create() system call failed
DBPLUS_ERR_LSEEK	ERR_LSEEK	Lseek() system call failed
DBPLUS_ERR_LENGTH	ERR_LENGTH	Tuple exceeds maximum length
DBPLUS_ERR_OPEN	ERR_OPEN	Open() system call failed
DBPLUS_ERR_WOPEN	ERR_WOPEN	Relation already opened for writing
DBPLUS_ERR_MAGIC	ERR_MAGIC	File is not a relation
DBPLUS_ERR_VERSION	ERR_VERSION	File is a very old relation

PHP Constant	db++ constant	meaning
DBPLUS_ERR_PGSIZE	ERR_PGSIZE	Relation uses a different page size
DBPLUS_ERR_CRC	ERR_CRC	Invalid crc in the superpage
DBPLUS_ERR_PIPE	ERR_PIPE	Piped relation requires lseek()
DBPLUS_ERR_NIDX	ERR_NIDX	Too many secondary indices
DBPLUS_ERR_MALLOC	ERR_MALLOC	Malloc() call failed
DBPLUS_ERR_NUSERS	ERR_NUSERS	Error use of max users
DBPLUS_ERR_PREEXIT	ERR_PREEXIT	Caused by invalid usage
DBPLUS_ERR_ONTRAP	ERR_ONTRAP	Caused by a signal
DBPLUS_ERR_PREPROC	ERR_PREPROC	Error in the preprocessor
DBPLUS_ERR_DBPARSE	ERR_DBPARSE	Error in the parser
DBPLUS_ERR_DBRUNERR	ERR_DBRUNERR	Run error in db
DBPLUS_ERR_DBPREEXIT	ERR_DBPREEXIT	Exit condition caused by prexit() * procedure
DBPLUS_ERR_WAIT	ERR_WAIT	Wait a little (Simple only)
DBPLUS_ERR_CORRUPT_TUPLE	ERR_CORRUPT_TUPLE	A client sent a corrupt tuple
DBPLUS_ERR_WARNING0	ERR_WARNING0	The Simple routines encountered a non fatal error which was corrected
DBPLUS_ERR_PANIC	ERR_PANIC	The server should not really die but after a disaster send ERR_PANIC to all its clients
DBPLUS_ERR_FIFO	ERR_FIFO	Can’t create a fifo
DBPLUS_ERR_PERM	ERR_PERM	Permission denied
DBPLUS_ERR_TCL	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED	ERR_RESTRICTED	Only two users
DBPLUS_ERR_USER	ERR_USER	An error in the use of the library by an application programmer
DBPLUS_ERR_UNKNOWN	ERR_UNKNOWN	

dbplus_add (PHP 4 >= 4.1.0)

Add a tuple to a relation

int **dbplus_add** (resource relation, array tuple)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

This function will add a tuple to a relation. The `tuple` data is an array of attribute/value pairs to be inserted into the given `relation`. After successful execution the `tuple` array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_aql (PHP 4 >= 4.1.0)

Perform AQL query

resource **dbplus_aql** (string query [, string server [, string dbpath]])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_aql() will execute an AQL `query` on the given `server` and `dbpath`.

On success it will return a relation handle. The result data may be fetched from this relation by calling dbplus_next() and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the original db++ manual.

dbplus_chdir (PHP 4 >= 4.1.0)

Get/Set database virtual current directory

string **dbplus_chdir** ([string newdir])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_chdir() will change the virtual current directory where relation files will be looked for by dbplus_open(). **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any `newdir` may be used to query the current working directory.

dbplus_close (PHP 4 >= 4.1.0)

Close a relation

int **dbplus_close** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Calling **dbplus_close()** will close a relation previously opened by dbplus_open().

dbplus_curr (PHP 4 >= 4.1.0)

Get current tuple from relation

int **dbplus_curr** (resource relation, array tuple)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_curr() will read the data for the current tuple for the given `relation` and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_errcode (PHP 4 >= 4.1.0)

Get error string for given errorcode or last error

string **dbplus_errcode** (int errno)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_errcode() returns a cleartext error string for the error code passed as `errno` of for the result code of the last db++ operation if no parameter is given.

dbplus_errno (PHP 4 >= 4.1.0)

Get error code for last operation

int **dbplus_errno** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_errno() will return the error code returned by the last db++ operation.

See also dbplus_errcode().

dbplus_find (PHP 4 >= 4.1.0)

Set a constraint on a relation

int **dbplus_find** (resource relation, array constraints, mixed tuple)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_find() will place a constraint on the given relation. Further calls to functions like dbplus_curr() or dbplus_next() will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The `constraints` parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also dbplus_unselect().

dbplus_first (PHP 4 >= 4.1.0)

Get first tuple from relation

int **dbplus_first** (resource relation, array tuple)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_curr() will read the data for the first tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_curr(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_flush (PHP 4 >= 4.1.0)

Flush all changes made on a relation

int **dbplus_flush** (resource relation)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_flush() will write all changes applied to `relation` since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks (PHP 4 >= 4.1.0)

Free all locks held by this client

int **dbplus_freealllocks** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_freeaalllocks() will free all tuple locks held by this client.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freerlocks().

dbplus_freelock (PHP 4 >= 4.1.0)

Release write lock on tuple

int **dbplus_freelock** (resource relation, string tname)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_freelock() will release a write lock on the given `tuple` previously obtained by dbplus_getlock().

See also dbplus_getlock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_freerlocks (PHP 4 >= 4.1.0)

Free all tuple locks on given relation

int **dbplus_freerlocks** (resource relation)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_freerlocks() will free all tuple locks held on the given `relation`.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freealllocks().

dbplus_getlock (PHP 4 >= 4.1.0)

Get a write lock on a tuple

int **dbplus_getlock** (resource relation, string tname)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_getlock() will request a write lock on the specified `tuple`. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also dbplus_freelock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_getunique (PHP 4 >= 4.1.0)

Get a id number unique to a relation

int **dbplus_getunique** (resource relation, int uniqueid)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_getunique() will obtain a number guaranteed to be unique for the given `relation` and will pass it back in the variable given as `uniqueid`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_info (PHP 4 >= 4.1.0)

???

int **dbplus_info** (resource relation, string key, array)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_last (PHP 4 >= 4.1.0)

Get last tuple from relation

int **dbplus_last** (resource relation, array tuple)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_curr() will read the data for the last tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_next().

dbplus_lockrel (unknown)

Request write lock on relation

int **dbplus_lockrel** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can't alter it while it is locked.

dbplus_next (PHP 4 >= 4.1.0)

Get next tuple from relation

int **dbplus_next** (resource relation, array)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_last().

dbplus_open (PHP 4 >= 4.1.0)

Open relation file

resource **dbplus_open** (string name)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

The relation file `name` will be opened. `name` can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commands referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling dbplus_errno().

dbplus_prev (PHP 4 >= 4.1.0)

Get previous tuple from relation

int **dbplus_prev** (resource relation, array tuple)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_next(), and dbplus_last().

dbplus_rchperm (PHP 4 >= 4.1.0)

Change relation permissions

int **dbplus_rchperm** (resource relation, int mask, string user, string group)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rchperm() will change access permissions as specified by `mask`, `user` and `group`. The values for these are operating system specific.

dbplus_rcreate (PHP 4 >= 4.1.0)

Creates a new DB++ relation

resource **dbplus_rcreate** (string name, mixed domlist [, boolean overwrite])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rcreate() will create a new relation named `name`. An existing relation by the same name will only be overwritten if the relation is currently not in use and `overwrite` is set to TRUE.

`domlist` should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain

description strings, but it is recommended to use an array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentation, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact (PHP 4 >= 4.1.0)

Creates an exact but empty copy of a relation including indices

resource **dbplus_rcrtexact** (string name, resource relation, boolean overwrite)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rcrtexact() will create an exact but empty copy of the given `relation` under a new `name`. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_rcrtlike (PHP 4 >= 4.1.0)

Creates an empty copy of a relation with default indices

resource **dbplus_rcrtlike** (string name, resource relation, int flag)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rcrtexact() will create an empty copy of the given `relation` under a new `name`, but with default indices. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_resolve (PHP 4 >= 4.1.0)

Resolve host information for relation

int **dbplus_resolve** (string relation_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_resolve() will try to resolve the given `relation_name` and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or `FALSE` on error.

See also dbplus_tcl().

dbplus_rkeys (PHP 4 >= 4.1.0)

Specify new primary key for a relation

resource **dbplus_rkeys** (resource relation, mixed domlist)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rkeys() will replace the current primary key for `relation` with the combination of domains specified by `domlist`.

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_restorepos (PHP 4 >= 4.1.0)

???

int **dbplus_restorepos** (resource relation, array tuple)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_ropen (PHP 4 >= 4.1.0)

Open relation file local

resource **dbplus_ropen** (string name)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_ropen() will open the relation `file` locally for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and dbplus_next() may be applied to the returned relation.

dbplus_rquery (PHP 4 >= 4.1.0)

Perform local (raw) AQL query

int **dbplus_rquery** (string query, string dbpath)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than dbplus_aql() but will work on local data only.

dbplus_rrename (PHP 4 >= 4.1.0)

Rename a relation

int **dbplus_rrename** (resource relation, string name)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rrename() will change the name of `relation` to `name`.

dbplus_rsecindex (PHP 4 >= 4.1.0)

Create a new secondary index for a relation

resource **dbplus_rsecindex** (resource relation, mixed domlist, int type)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rsecindex() will create a new secondary index for `relation` with consists of the domains specified by `domlist` and is of type `type`

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_runlink (PHP 4 >= 4.1.0)

Remove relation from filesystem

int **dbplus_runlink** (resource relation)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_unlink() will close and remove the `relation`.

dbplus_rzap (PHP 4 >= 4.1.0)

Remove all tuples from relation

int **dbplus_rzap** (resource relation)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_rzap() will remove all tuples from `relation`.

dbplus_savepos (PHP 4 >= 4.1.0)

???

int **dbplus_savepos** (resource relation)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_setindex (PHP 4 >= 4.1.0)

???

int **dbplus_setindex** (resource relation, string idx_name)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_setindexbynumber (PHP 4 >= 4.1.0)

???

int **dbplus_setindexbynumber** (resource relation, int idx_number)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_sql (PHP 4 >= 4.1.0)

Perform SQL query

resource **dbplus_sql** (string query, string server, string dbpath)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_tcl (PHP 4 >= 4.1.0)

Execute TCL code on server side

int **dbplus_tcl** (int sid, string script)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL `script` code should be executed by. dbplus_resolve() will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also dbplus_resolve().

dbplus_tremove (PHP 4 >= 4.1.0)

Remove tuple and return new current tuple

int **dbplus_tremove** (resource relation, array tuple [, array current])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_tremove() removes `tuple` from `relation` if it perfectly matches a tuple within the relation. `current`, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo (PHP 4 >= 4.1.0)

???

int **dbplus_undo** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_undoprepare (PHP 4 >= 4.1.0)

???

int **dbplus_undoprepare** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Not implemented yet.

dbplus_unlockrel (PHP 4 >= 4.1.0)

Give up write lock on relation

int **dbplus_unlockrel** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_unlockrel() will release a write lock previously obtained by dbplus_lockrel().

dbplus_unselect (PHP 4 >= 4.1.0)

Remove a constraint from relation

int **dbplus_unselect** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Calling **dbplus_unselect()** will remove a constraint previously set by dbplus_find() on `relation`.

dbplus_update (PHP 4 >= 4.1.0)

Update specified tuple in relation

int **dbplus_update** (resource relation, array old, array new)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_update() replaces the tuple given by `old` with the data from `new` if and only if `old` completely matches a tuple within `relation`.

dbplus_xlockrel (PHP 4 >= 4.1.0)

Request exclusive lock on relation

int **dbplus_xlockrel** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_xlockrel() will request an exclusive lock on `relation` preventing even read access from other clients.

See also dbplus_xunlockrel().

dbplus_xunlockrel (PHP 4 >= 4.1.0)

Free exclusive lock on relation

int **dbplus_xunlockrel** (resource relation)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

dbplus_xunlockrel() will release an exclusive lock on `relation` previously obtained by dbplus_xlockrel().

XXIII. Direkte Ein-/Ausgabe Funktionen

Direkte Ein-/Ausgabe Funktionen

PHP unterstützt die direkte Ein-/Ausgabe-Funktionen, beschrieben im Posix Standard (Kapitel 6) für die Ein- und Ausgabe auf einer niederen Ebene als die standardmässig verwendeten C-Funktionen (fopen, fread, ...).

Installation

Um die DIO Funktionen benutzen zu können, muss PHP mit der Option `--enable-dio` konfiguriert werden.

dio_open (PHP 4 CVS only)

Öffnet eine Datei mit den angegebenen Rechten

resource **dio_open** (string filename, int flags [, int mode])

dio_open() öffnet eine Datei und liefert einen Dateizeiger zurück. Im Fehlerfall liefert **dio_open()** -1 zurück. Ist der Parameter *flags* O_CREAT, so gibt der optionale Parameter *mode* die Dateirechte an. Der Parameter *flags* kann eine der folgenden Zugriffsmethoden sein:

- O_RDONLY - öffnet die Datei für Lesezugriff
- O_WRONLY - öffnet die Datei für Schreibzugriff
- O_RDWR - öffnet die Datei für Schreib- und Lesezugriff

Diese Zugriffsmethoden können mit folgenden Attributen kombiniert werden:

- O_CREAT - erzeugt die Datei, falls diese nicht existiert
- O_EXCL - in Kombination mit O_CREAT wird ein Fehler (-1) zurückgegeben, wenn die Datei bereits existiert
- O_TRUNC - falls die Datei bereits existiert, und sie für Schreibzugriff geöffnet wird, wird diese auf die Grösse 0 zurückgesetzt
- O_APPEND - setzt den Zeiger für Schreiboperationen an das Ende der Datei
- O_NONBLOCK - Setzt den Modus auf "nonblocking"

dio_read (PHP 4 CVS only)

Liest n Bytes und gibt sie zurück, ist n nicht angegeben wird ein 1K Block gelesen

string **dio_read** (resource fd [, int n])

Die Funktion **dio_read()** liest *n* Bytes aus der Datei *fd* und gibt diese zurück. Wird der optionale Parameter *n* nicht angegeben, wird standardmässig ein 1K Block gelesen und zurückgegeben.

dio_write (PHP 4 CVS only)

Schreibt Daten in eine Datei

int **dio_write** (resource fd, string data [, int len])

Die Funktion **dio_write()** schreibt bis zu *len* Bytes an Daten *data* in die Datei *fd*. Wird der optionale Parameter *len* nicht angegeben, so werden alle Daten *data* in die Datei geschrieben. **dio_write()** liefert die Anzahl der Bytes zurück, die in die Datei geschrieben wurden.

dio_truncate (PHP 4 CVS only)

Schneidet eine Datei ab

bool **dio_truncate** (resource fd, int offset)

Die Funktion **dio_truncate()** schneidet die Datei *fd* nach *offset* ab. War die Datei ursprünglich grösser, so gehen die Daten verloren. War sie kleiner so wird sie mit Nullen aufgefüllt.

dio_stat (PHP 4 CVS only)

Liefert Statusinformationen über eine Datei

array **dio_stat** (resource fd)

Die Funktion fstat() gibts ein assoziatives Array zurück, welches Informationen über die Datei *fd* enthält. Das Array enthält folgende Einträge:

- "device" - Device
- "inode" - Inode
- "mode" - Modus
- "nlink" - Anzahl Links
- "uid" - Benutzer Id
- "gid" - Gruppentyp
- "device_type" - Device Typ
- "size" - Grösse in Bytes
- "blocksize" - Blockgrösse
- "blocks" - Anzahl allocierter Blöcke
- "atime" - Letzter Zugriff
- "mtime" - Letze Modifikation
- "ctime" - Letzte Änderung

Im Fehlerfall liefert **dio_stat()** NULL zurück.

dio_seek (PHP 4 CVS only)

Setzt den Dateizeiger

int **dio_seek** (resource fd, int pos, int whence)

Die Funktion **dio_seek()** setzt den Dateizeiger der Datei *fd*. Der *whence* gibt dabei an, wie die Position *pos* interpretiert werden soll:

- SEEK_SET - gibt an, dass *pos* die Position vom Anfang der Datei ist
- SEEK_CUR - gibt an, dass *pos* die Position relativ von der momentanen Dateizeigerposition ist.

- SEEK_END - gibt an, dass `pos` die Position relativ zum Dateiende ist. Eine negative Zahl setzt den Dateizeiger innerhalb der Datei, ein positvier Wert bewirkt, dass die Datei vom Ende bis zur neuen Position mit Nullen aufgefüllt wird.

dio_fcntl (PHP 4 CVS only)

Ändert den Dateideskriptor

mixed **dio_fcntl** (resource fd, int cmd [, mixed arg])

Die Funktion **dio_fcntl()** führt eine Aktion `cmd` an der Datei `fd` aus. Einige Aktionen `cmd` benötigen das zusätzliche Argument `args`.

`arg` ist ein assoziatives Array, wenn `cmd` F_SETLK oder F_SETLLW ist. Es hat folgende Einträge:

- "start" - offset an dem der Lock beginnt
- "length" - Länge des gelockten Bereichs
- "wenth" - Relative Position von start. Kann folgende Werte annehmen: SEEK_SET, SEEK_END and SEEK_CUR
- "type" - Locktyp: F_RDLCK (read lock), F_WRLCK (write lock) oder F_UNLCK (unlock)

`cmd` ist einer der folgenden Konstanten:

- F_SETLK - Setzt oder löscht einen Lock. Ist der Bereich bereits von einem anderen Prozess gelockt, so liefert die Funktion **dio_fcntl()** -1 zurück.
- F_SETLKW - wie F_SETLK, jedoch im Falle, dass der Bereich von einem anderen Prozess gelockt wird, wartet **dio_fcntl()** auf die Freigabe des Locks.
- F_GETLK - **dio_fcntl()** liefert ein assioziatives Array zurück (Beschreibung siehe oben), falls ein anderer Prozess den Bereich gelockt hat.

dio_close (PHP 4 CVS only)

Schliesst die Datei

void **dio_close** (resource fd)

Die Funktion **dio_close()** schliesst die Datei `fd`.

XXIV. Verzeichnis-Funktionen

chdir (PHP 3, PHP 4 >= 4.0.0)

Wechseln des Verzeichnisses

int **chdir** (string directory)

Wechsel des aktuellen PHP-Verzeichnisses in das durch *`directory`* angegebene Verzeichnis. Gibt `FALSE` zurück, wenn der Wechsel nicht möglich war, sonst `TRUE`.

dir (PHP 3, PHP 4 >= 4.0.0)

Verzeichnis-Klasse

```
 class dir {
  dir(string directory);
  string path;
  string read();
  void rewind();
  void close();
}
```

new **dir** (string directory)

Ein Pseudo-Objekt-Orientierter Mechanismus zum Lesen eines Verzeichnisses. Das per *`directory`* angegebene Verzeichnis wird geöffnet. Sobald das Verzeichnis geöffnet ist, stehen 2 Eigenschaften zur Verfügung. Die Eigenschaft Handle kann mit anderen Directory-Funktionen wie readdir(), rewinddir() und closedir() benutzt werden (Handles sind Verweise auf Einträge in einer internen Tabelle zur Verwaltung von Objekten). Die Eigenschaft Path enthält die Pfadangabe des geöffneten Verzeichnisses. Es sind 3 Methoden verfügbar: read, rewind und close.

Beispiel 1. dir()-Beispiel:

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while($entry=$d->read()) {
    echo $entry."<br>\n";
}
$d->close();
```

closedir (PHP 3, PHP 4 >= 4.0.0)

Beenden eines Verzeichnis-Handles

void **closedir** (int dir_handle)

Beendet den mit *`dir_handle`* angegebenen Verzeichnis-Stream (Ein-/Ausgabe-Kanal für Verzeichnis-Objekte). Dieser muss vorher durch opendir() geöffnet worden sein.

getcwd (PHP 4 >= 4.0.0)

Ermittelt das aktuelle Arbeitsverzeichnis

string **getcwd** (void)

Gibt das gegenwärtige Arbeitsverzeichnis zurück.

opendir (PHP 3, PHP 4 >= 4.0.0)

Öffnen eines Verzeichnis-Handles

int **opendir** (string path)

Gibt ein Verzeichnis-Handle zurück, das in späteren Aufrufen von closedir(), readdir() und rewinddir() gebraucht werden kann.

readdir (PHP 3, PHP 4 >= 4.0.0)

Liest den Eintrag eines Verzeichnis-Handles

string **readdir** (int dir_handle)

Ermittelt den nächsten Dateinamen im Verzeichnis. Die Dateinamen werden ungeordnet zurück gegeben.

Beispiel 1. Liste alle Dateien des aktuellen Verzeichnisses auf

```
<?php
$handle=opendir ('.');
echo "Verzeichnis-Handle: $handle\n";
echo "Dateien:\n";
while (false !== ($file = readdir ($handle))) {
    echo "$file\n";
}
closedir($handle);
?>
```

Beachten sie, dass **readdir()** auch die Einträge "." und ".." zurück gibt. Wollen sie das nicht, müssen sie diese ausschliessen:

Beispiel 2. Liste alle Dateien des aktuellen Verzeichnisses unter Ausschluss von "." und ".." auf

```
<?php
$handle=opendir('.');
while ($file = readdir ($handle)) {
    if ($file != "." && $file != "..") {
        echo "$file\n";
```

```
    }
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4 >= 4.0.0)

Zurücksetzen des Verzeichnis-Handles

void **rewinddir** (int dir_handle)

Setzt den mit *dir_handle* angegebenen Verzeichnis-Stream auf den Anfang zurück (Reset).

XXV. DOM XML Funktionen

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

Diese Funktionen sind nur verfügbar, wenn PHP mit der Option `--with-dom=[DIR]` konfiguriert und kompiliert wurde. Hierfür wird mindestens libxml-2.0.0 benötigt (die Beta-Version reicht nicht). Diese Funktionen wurden in PHP4 eingeführt.

Dieses Modul definiert die folgenden Konstanten:

Tabelle 1. XML Konstanten

Konstante	Wert	Beschreibung
XML_ELEMENT_NODE	1	
XML_ATTRIBUTE_NODE	2	
XML_TEXT_NODE	3	
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	
XML_PI_NODE	7	
XML_COMMENT_NODE	8	
XML_DOCUMENT_NODE	9	
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

Dieses Modul definiert eine Reihe von Klassen. Die DOM XML Funktionen geben einen Parsebaum eines XML Dokumentes zurück, dessen Elemente jeweils einer der nachstehenden Klassen angehören.

xmldoc (PHP 4 >= 4.0.0)

Erzeugt ein Objekt mit der DOM Repräsentation eines XML Dokumentes

object **xmldoc** (string str)

Die Funktion verarbeitet das XML Dokument in `str` und liefert eine Instanz der Klasse "Dom document", mit den Eigenschaften "doc" (resource), "version" (string) und "type" (long).

xmldocfile (PHP 4 >= 4.0.0)

Erzeugt ein Objekt mit der DOM Repräsentation einer XML Datei

object **xmldocfile** (string filename)

Die Funktion liest die XML Datei in `filename` und liefert eine Instanz der Klasse "Dom document", mit den Eigenschaften "doc" (resource), "version" (string) und "type" (long).

xmltree (PHP 4 >= 4.0.0)

Erzeugt eine Baum-Repräsentation eines XML Dokumentes

object **xmltree** (string str)

Die Funktion verarbeitet das XML Dokument in `str` und liefert einen Baum von PHP Objekten.

XXVI. .NET Funktionen

dotnet_load (unknown)

Lädt ein DOTNET Modul

int **dotnet_load** (string assembly_name [, string datatype_name [, int codepage]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XXVII. Error Handling and Logging Functions

These are functions dealing with error handling and logging. They allow you to define your own error handling rules, as well as modify the way the errors can be logged. This allows you to change and enhance error reporting to suit your needs.

With the logging functions, you can send messages directly to other machines, to an email (or email to pager gateway!), to system logs, etc., so you can selectively log and monitor the most important parts of your applications and websites.

The error reporting functions allow you to customize what level and kind of error feedback is given, ranging from simple notices to customized functions returned during errors.

error_log (PHP 3, PHP 4 >= 4.0.0)

Sendet eine Fehlermeldung

int **error_log** (string message, int message_type [, string destination [, string extra_headers]])

Sendet eine Fehlermeldung an das Log-File des Webservers, einen TCP Port oder eine Datei. Der erste Parameter, `message`, ist die Fehlermeldung, die mitgeschrieben werden soll. Der zweite Parameter, `message_type` gibt an, wo die Meldung abgelegt werden soll:

Tabelle 1. error_log() Typem

0	`Message` wird mit Hilfe des internen Log-Systems des Betriebssystemes oder einer Datei geschrieben, je nachdem, wie error_log eingestellt ist.
1	`Message` wird via Email an die Adresse gesendet, die mit `destination` amnngegeben wurde. Dies ist der einzige Punkt, bei dem der vierte Parameter `extra_headers` verwendet wird. Die Nachricht verwendet die selbe interne Funktion wie mail().
2	`Message` wird über den PHP-Debugger gesendet. Diese Option ist nur verfügbar, wenn der Debugger gestartet wurde. Ist das der Fall, so wird mit dem `destination` Parameter der Hostname oder die IP-Adresse gesetzt. Optional auch der Port, unter dem der Debugger läuft.
3	`Message` wird an die Datei angefügt, die mit `destination` angegeben wurde.

Beispiel 1. error_log() Beispiele

```
// Server sendet eine Nachricht an den Administrator,
// daß keine Verbindung zur Datenbank möglich ist.
if (!Ora_Logon($username, $password)) {
    error_log ("Oracle database not available!", 0);
}

// Server sendet eine Nachricht an den Administrator, daß FOO langsam ausgeht
if (!($foo = allocate_new_foo()) {
    error_log ("Big trouble, we're all out of FOOs!", 1,
               "operator@mydomain.com");
}

// AndereMöglichkeiten, error_log() aufzurufen:
error_log ("You messed up!", 2, "127.0.0.1:7000");
error_log ("You messed up!", 2, "loghost");
error_log ("You messed up!", 3, "/var/tmp/my-errors.log");
```

error_reporting (PHP 3, PHP 4 >= 4.0.0)

Gibt an, wie PHP-Fehlermeldungen gezeigt werden

int **error_reporting** ([int level])

Setzt die Stufe von PHP's Fehlermeldungen. Diese Stufe ist der Fehlermeldung hängt von den folgenden Werten ab. (Hinter den folgenden Links verbirgt sich die Bedeutung der internen Werte):

Tabelle 1. error_reporting() Werte

Werte	interne Namen
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING

restore_error_handler (PHP 4)

Restores the previous error handler function

void **restore_error_handler** (void)

Used after changing the error handler function using set_error_handler(), to revert to the previous error handler (which could be the built-in or a user defined function)

See also error_reporting(), set_error_handler(), trigger_error(), user_error()

set_error_handler (PHP 4)

Sets a user-defined error handler function.

string **set_error_handler** (string error_handler)

Sets a user function (*error_handler*) to handle errors in a script. Returns the previously defined error handler (if any), or FALSE on error. This function can be used for defining your own way of handling errors during runtime, for example in applications in which you need to do cleanup of data/files when a critical error happens, or when you need to trigger an error under certain conditions (using trigger_error())

The user function needs to accept 2 parameters: the error code, and a string describing the error. The example below shows the handling of internal execptions by triggering errors and handling them with a user defined function:

Beispiel 1. Error handling with set_error_handler() and trigger_error()

```
<?php

// redefine the user error constants - PHP4 only
define(FATAL,E_USER_ERROR);
define(ERROR,E_USER_WARNING);
define(WARNING,E_USER_NOTICE);

// set the error reporting level for this script
error_reporting(FATAL + ERROR + WARNING);

// error handler function
function myErrorHandler ($errno, $errstr) {
    switch ($errno) {
    case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".__LINE__." of file ".__FILE__;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit -1;
    break;
    case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
    case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
    }
}

// function to test the error handling
function scale_by_log ($vect, $scale) {
    if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale",
      FATAL);
    if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values expected", ERROR);
    return null;
    }
    for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
    trigger_error("Value at position $i is not a number, using 0 (zero)",
      WARNING);
    $temp[$i] = log($scale) * $vect[$i];
    }
    return $temp;
}

// set to the user defined error handler
```

```
$old_error_handler = set_error_handler("myErrorHandler");

// trigger some errors, first define a mixed array with a non-numeric item
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// now generate second array, generating a warning
echo "----\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// this is trouble, we pass a string instead of an array
echo "----\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// this is a critical error, log of zero or negative number is undefined
echo "----\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);

?>
```

And when you run this sample script, the output will be

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, using 0 (zero)<br>
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br>
NULL
----
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br>
  Fatal error in line 16 of file trigger_error.php, PHP 4.0.1pl2 (Linux)<br>
Aborting...<br>
```

See also error_reporting(), restore_error_handler(), trigger_error(), user_error()

trigger_error (PHP 4)

Generates a user-level error/warning/notice message

void **trigger_error** (string error_msg [, int error_type])

Used to trigger a user error condition, it can be used by in conjunction with the built-in error handler, or with a user defined function that has been set as the new error handler (set_error_handler()). This function is useful when you need to generate a particular response to an exception at runtime. For example:

```
if (assert($divisor == 0))
   trigger_error("Cannot divide by zero", E_USER_ERROR);
```

Anmerkung: See set_error_handler() for a more extensive example.

See also error_reporting(), set_error_handler(), restore_error_handler(), user_error()

user_error (PHP 4 >= 4.0.0)

Generates a user-level error/warning/notice message

void **user_error** (string error_msg [, int error_type])

This is an alias for the function trigger_error().

See also error_reporting(), set_error_handler(), restore_error_handler(), trigger_error()

XXVIII. FrontBase Functions

These functions allow you to access FrontBase database servers. In order to have these functions available, you must compile php with fbsql support by using the `--with-fbsql` option. If you use this option without specifying the path to fbsql, php will search for the fbsql client libraries in the default installation location for the platform. Users who installed FrontBase in a non standard directory should always specify the path to fbsql: `--with-fbsql=/path/to/fbsql`. This will force php to use the client libraries installed by FrontBase, avoiding any conflicts.

More information about FrontBase can be found at http://www.frontbase.com/.

Documentation for FrontBase can be found at http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Frontbase support has been added to PHP 4.0.6.

fbsql_affected_rows (PHP 4 >= 4.0.6)

Get number of affected rows in previous FrontBase operation

int **fbsql_affected_rows** ([int link_identifier])

fbsql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *link_identifier*. If the link identifier isn't specified, the last link opened by fbsql_connect() is assumed.

> **Anmerkung:** If you are using transactions, you need to call **fbsql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Anmerkung:** When using UPDATE, FrontBase will not update columns where the new value is the same as the old value. This creates the possibility that **fbsql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

If the last query failed, this function will return -1.

See also: fbsql_num_rows().

fbsql_autocommit (PHP 4 >= 4.0.6)

Enable or disable autocommit.

bool **fbsql_autocommit** (resource link_identifier [, bool OnOff])

fbsql_autocommit() returns the current autocommit status. if the optional OnOff parameter is given the auto commit status will be changed. With OnOff set to `TRUE` each statement will be committed automatically, if no errors was found. With OnOff set to `FALSE` the user must commit or rollback the transaction using either fbsql_commit() or fbsql_rollback().

See also: fbsql_commit() and fbsql_rollback()

fbsql_change_user (unknown)

Change logged in user of the active connection

resource **fbsql_change_user** (string user, string password [, string database [, int link_identifier]])

fbsql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

fbsql_close (PHP 4 >= 4.0.6)

Close FrontBase connection

boolean **fbsql_close** ([resource link_identifier])

Returns: TRUE on success, FALSE on error.

fbsql_close() closes the connection to the FrontBase server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **fbsql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Beispiel 1. fbsql_close() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);
?>
```

See also: fbsql_connect() and fbsql_pconnect().

fbsql_commit (PHP 4 >= 4.0.6)

Commits a transaction to the database

bool **fbsql_commit** ([resource link_identifier])

Returns: TRUE on success, FALSE on failure.

fbsql_commit() ends the current transaction by writing all inserts, updates and deletes to the disk and unlocking all row and table locks held by the transaction. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_rollback()

fbsql_connect (PHP 4 >= 4.0.6)

Open a connection to a FrontBase Server

resource **fbsql_connect** ([string hostname [, string username [, string password]]])

Returns a positive FrontBase link identifier on success, or an error message on failure.

fbsql_connect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *hostname* = 'NULL', *username* = '_SYSTEM' and *password* = empty password.

If a second call is made to **fbsql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling fbsql_close().

Beispiel 1. fbsql_connect() example

```
<?php

    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);

?>
```

See also fbsql_pconnect() and fbsql_close().

fbsql_create_db (PHP 4 >= 4.0.6)

Create a FrontBase database

bool **fbsql_create_db** (string database name [, resource link_identifier])

fbsql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Beispiel 1. fbsql_create_db() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    if (fbsql_create_db ("my_db")) {
        print("Database created successfully\n");
    } else {
        printf("Error creating database: %s\n", fbsql_error ());
    }
?>
```

See also: fbsql_drop_db().

fbsql_create_blob (PHP 4 CVS only)

Create a BLOB

string **fbsql_create_blob** (string blob_data [, resource link_identifier])

Returns: A resource handle to the newly created blob.

fbsql_create_blob() creates a blob from blob_data. The returned resource handle can be used with insert and update commands to store the blob in the database.

Beispiel 1. fbsql_create_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "blobfile.bin";
    $fp = fopen($filename, "rb");
    $blobdata = fread($fp, filesize($filename));
    fclose($fp);

    $blobHandle = fbsql_create_blob($blobdata, $link);

    $sql = "INSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_clob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_clob (PHP 4 CVS only)

Create a CLOB

string **fbsql_create_clob** (string clob_data [, resource link_identifier])

Returns: A resource handle to the newly created CLOB.

fbsql_create_clob() creates a clob from clob_data. The returned resource handle can be used with insert and update commands to store the clob in the database.

Beispiel 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "clob_file.txt";
    $fp = fopen($filename, "rb");
    $clobdata = fread($fp, filesize($filename));
    fclose($fp);

    $clobHandle = fbsql_create_clob($clobdata, $link);

    $sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_database_password (PHP 4 >= 4.0.6)

Sets or retrieves the password for a FrontBase database

string **fbsql_database_password** (resource link_identifier [, string database_password])

Returns: The database password associated with the link identifier.

fbsql_database_password() sets and retrieves the database password used by the connection. if a database is protected by a database password, the user must call this function before calling fbsql_select_db(). if the second optional parameter is given the function sets the database password for the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

This function does not change the database password in the database nor can it be used to retrive the database password for a database.

Beispiel 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    fbsql_database_password($link, "secret db password");
    fbsql_select_db($database, $link);
?>
```

See also: fbsql_connect(), fbsql_pconnect() and fbsql_select_db().

fbsql_data_seek (PHP 4 >= 4.0.6)

Move internal result pointer

bool **fbsql_data_seek** (resource result_identifier, int row_number)

Returns: `TRUE` on success, `FALSE` on failure.

fbsql_data_seek() moves the internal row pointer of the FrontBase result associated with the specified result identifier to point to the specified row number. The next call to fbsql_fetch_row() would return that row.

`Row_number` starts at 0.

Beispiel 1. fbsql_data_seek() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");

    fbsql_select_db ("samp_db")
        or die ("Could not select database");

    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = fbsql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = fbsql_fetch_object ($result)))
            continue;

        printf("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    fbsql_free_result ($result);
?>
```

fbsql_db_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_db_query** (string database, string query [, resource link_identifier])

Returns: A positive FrontBase result identifier to the query result, or `FALSE` on error.

fbsql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the FrontBase server and if no such link is found it'll try to create one as if fbsql_connect() was called with no arguments

See also fbsql_connect().

fbsql_db_status (PHP 4 >= 4.1.0)

Get the status for a given database

int **fbsql_db_status** (string database_name [, resource link_identifier])

Returns: An integer value with the current status.

fbsql_db_status() requests the current status of the database specified by *database_name*. If the *link_identifier* is omitted the default link_identifier will be used.

The return value can be one of the following constants:

- FALSE - The exec handler for the host was invalid. This error will occur when the link_identifier connects directly to a database by using a port number. FBExec can be available on the server but no connection has been made for it.
- FBSQL_UNKNOWN - The Status is unknown.
- FBSQL_STOPPED - The database is not running. Use fbsql_start_db() to start the database.
- FBSQL_STARTING - The database is starting.
- FBSQL_RUNNING - The database is running and can be used to perform SQL operations.
- FBSQL_STOPPING - The database is stopping.
- FBSQL_NOEXEC - FBExec is not running on the server and it is not possible to get the status of the database.

See also: fbsql_start_db() and fbsql_stop_db().

fbsql_drop_db (PHP 4 >= 4.0.6)

Drop (delete) a FrontBase database

bool **fbsql_drop_db** (string database_name [, resource link_identifier])

Returns: TRUE on success, FALSE on failure.

fbsql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

fbsql_errno (PHP 4 >= 4.0.6)

Returns the numerical value of the error message from previous FrontBase operation

int **fbsql_errno** ([resource link_identifier])

Returns the error number from the last fbsql function, or 0 (zero) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed fbsql function (not including fbsql_error() and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
```

```
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_error() and fbsql_warnings().

fbsql_error (PHP 4 >= 4.0.6)

Returns the text of the error message from previous FrontBase operation

string **fbsql_error** ([resource link_identifier])

Returns the error text from the last fbsql function, or ″ (the empty string) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed fbsql function (not including **fbsql_error()** and fbsql_errno()), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_errno() and fbsql_warnings().

fbsql_fetch_array (PHP 4 >= 4.0.6)

Fetch a result row as an associative array, a numeric array, or both

array **fbsql_fetch_array** (resource result [, int result_type])

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_array() is an extended version of fbsql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **fbsql_fetch_array()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

The optional second argument *result_type* in **fbsql_fetch_array()** is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

For further details, see also fbsql_fetch_row() and fbsql_fetch_assoc().

Beispiel 1. fbsql_fetch_array() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select user_id, fullname from table");
while ($row = fbsql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_assoc (PHP 4 >= 4.0.6)

Fetch a result row as an associative array

array **fbsql_fetch_assoc** (resource result)

Returns an associative array that corresponds to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_assoc() is equivalent to calling fbsql_fetch_array() with FBSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way fbsql_fetch_array() originally worked. If you need the numeric indices as well as the associative, use fbsql_fetch_array().

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use fbsql_fetch_array() and have it return the numeric indices as well.

An important thing to note is that using **fbsql_fetch_assoc()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

For further details, see also fbsql_fetch_row() and fbsql_fetch_array().

Beispiel 1. fbsql_fetch_assoc() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select * from table");
while ($row = fbsql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_field (PHP 4 >= 4.0.6)

Get column information from a result and return as an object

object **fbsql_fetch_field** (resource result [, int field_offset])

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be `NULL`
- type - the type of the column

Beispiel 1. fbsql_fetch_field() example

```
<?php
fbsql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = fbsql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = fbsql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
```

```
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
}
fbsql_free_result ($result);
?>
```

See also fbsql_field_seek().

fbsql_fetch_lengths (PHP 4 >= 4.0.6)

Get the length of each output in a result

array **fbsql_fetch_lengths** ([resource result])

Returns: An array that corresponds to the lengths of each field in the last row fetched by fbsql_fetch_row(), or `FALSE` on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by fbsql_fetch_row(), fbsql_fetch_array() and fbsql_fetch_object() in an array, starting at offset 0.

See also: fbsql_fetch_row().

fbsql_fetch_object (PHP 4 >= 4.0.6)

Fetch a result row as an object

object **fbsql_fetch_object** (resource result [, int result_type])

Returns an object with properties that correspond to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_object() is similar to fbsql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *`result_type`* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to fbsql_fetch_array(), and almost as quick as fbsql_fetch_row() (the difference is insignificant).

Beispiel 1. fbsql_fetch_object() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database", "select * from table");
while ($row = fbsql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result ($result);
?>
```

See also: fbsql_fetch_array() and fbsql_fetch_row().

fbsql_fetch_row (PHP 4 >= 4.0.6)

Get a result row as an enumerated array

array **fbsql_fetch_row** (resource result)

Returns: An array that corresponds to the fetched row, or `FALSE` if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

See also: fbsql_fetch_array(), fbsql_fetch_object(), fbsql_data_seek(), fbsql_fetch_lengths(), and fbsql_result().

fbsql_field_flags (PHP 4 >= 4.0.6)

Get the flags associated with the specified field in a result

string **fbsql_field_flags** (resource result, int field_offset)

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

fbsql_field_name (PHP 4 >= 4.0.6)

Get the name of the specified field in a result

string **fbsql_field_name** (resource result, int field_index)

fbsql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Anmerkung:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Beispiel 1. fbsql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

fbsql_field_len (PHP 4 >= 4.0.6)

Returns the length of the specified field

int **fbsql_field_len** (resource result, int field_offset)

fbsql_field_len() returns the length of the specified field.

fbsql_field_seek (PHP 4 >= 4.0.6)

Set result pointer to a specified field offset

bool **fbsql_field_seek** (resource result, int field_offset)

Seeks to the specified field offset. If the next call to fbsql_fetch_field() doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: fbsql_fetch_field().

fbsql_field_table (PHP 4 >= 4.0.6)

Get name of the table the specified field is in

string **fbsql_field_table** (resource result, int field_offset)

Returns the name of the table that the specified field is in.

fbsql_field_type (PHP 4 >= 4.0.6)

Get the type of the specified field in a result

string **fbsql_field_type** (resource result, int field_offset)

fbsql_field_type() is similar to the fbsql_field_name() function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the FrontBase documentation (http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation).

Beispiel 1. fbsql_field_type() example

```
<?php

fbsql_connect ("localhost", "_SYSTEM", "");
fbsql_select_db ("wisconsin");
$result = fbsql_query ("SELECT * FROM onek;");
$fields = fbsql_num_fields ($result);
$rows   = fbsql_num_rows ($result);
$i = 0;
$table = fbsql_field_table ($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = fbsql_field_type  ($result, $i);
    $name  = fbsql_field_name  ($result, $i);
    $len   = fbsql_field_len   ($result, $i);
    $flags = fbsql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result (PHP 4 >= 4.0.6)

Free result memory

bool **fbsql_free_result** (int result)

fbsql_free_result() will free all memory associated with the result identifier *result*.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_insert_id (PHP 4 >= 4.0.6)

Get the id generated from the previous INSERT operation

int **fbsql_insert_id** ([resource link_identifier])

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns 0 if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Anmerkung:** The value of the FrontBase SQL function `LAST_INSERT_ID()` always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs (PHP 4 >= 4.0.6)

List databases available on a FrontBase server

resource **fbsql_list_dbs** ([resource link_identifier])

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the fbsql_tablename() function to traverse this result pointer.

Beispiel 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
```

```
    echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

Anmerkung: The above code would just as easily work with fbsql_fetch_row() or other similar functions.

fbsql_list_fields (PHP 4 >= 4.0.6)

List FrontBase result fields

resource **fbsql_list_fields** (string database_name, string table_name [, resource link_identifier])

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with fbsql_field_flags(), fbsql_field_len(), fbsql_field_name(), and fbsql_field_type().

A result identifier is a positive integer. The function returns -1 if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@fbsql()` then this error string will also be printed out.

Beispiel 1. fbsql_list_fields() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

fbsql_list_tables (PHP 4 >= 4.0.6)

List tables in a FrontBase database

resource **fbsql_list_tables** (string database [, resource link_identifier])

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result (PHP 4 >= 4.0.6)

Move the internal result pointer to the next result

bool **fbsql_next_result** (int result_id)

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the words from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` otherwise.

Beispiel 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
    fbsql_free_result($rs);
    fbsql_close ($link);
?>
```

fbsql_num_fields (PHP 4 >= 4.0.6)

Get number of fields in result

int **fbsql_num_fields** (resource result)

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), and fbsql_num_rows().

fbsql_num_rows (PHP 4 >= 4.0.6)

Get number of rows in result

int **fbsql_num_rows** (resource result)

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use fbsql_affected_rows().

Beispiel 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: fbsql_affected_rows(), fbsql_connect(), fbsql_select_db(), and fbsql_query().

fbsql_pconnect (PHP 4 >= 4.0.6)

Open a persistent connection to a FrontBase Server

resource **fbsql_pconnect** ([string hostname [, string username [, string password]]])

Returns: A positive FrontBase persistent link identifier on success, or `FALSE` on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *`host`* = 'localhost', *`username`* = "_SYSTEM" and *`password`* = empty password.

fbsql_pconnect() acts very much like fbsql_connect() with two major differences.

To set Frontbase server port number, use fbsql_select_db().

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called 'persistent'.

fbsql_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_query** (string query [, resource link_identifier])

fbsql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if fbsql_connect() was called with no arguments, and use it.

Anmerkung: The query string shall always end with a semicolon.

fbsql_query() returns `TRUE` (non-zero) or `FALSE` to indicate whether or not the query succeeded. A return value of `TRUE` means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns `FALSE`:

Beispiel 1. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **fbsql_query()** fails and returns `FALSE`:

Beispiel 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
```

```
?>
```

fbsql_query() will also fail and return `FALSE` if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call fbsql_num_rows() to find out how many rows were returned for a SELECT statement or fbsql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to fbsql_result(). When you are done with the result set, you can free the resources associated with it by calling fbsql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: fbsql_affected_rows(), fbsql_db_query(), fbsql_free_result(), fbsql_result(), fbsql_select_db(), and fbsql_connect().

fbsql_read_blob (PHP 4 CVS only)

Read a BLOB from the database

string **fbsql_read_blob** (string blob_handle [, resource link_identifier])

Returns: A string containing the BLOB specified by blob_handle.

fbsql_read_blob() reads BLOB data from the database. If a select statement contains BLOB and/or BLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_blob()** to get the actual BLOB data from the database.

Beispiel 1. fbsql_read_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the blob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the BLOB data in the first row
    $blob_data = fbsql_read_blob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), **fbsql_read_blob()**, fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_read_clob (PHP 4 CVS only)

Read a CLOB from the database

string **fbsql_read_clob** (string clob_handle [, resource link_identifier])

Returns: A string containing the CLOB specified by clob_handle.

fbsql_read_clob() reads CLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_clob()** to get the actual CLOB data from the database.

Beispiel 1. fbsql_read_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the clob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the CLOB data in the first row
    $clob_data = fbsql_read_clob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), **fbsql_read_clob()**, and fbsql_set_lob_mode().

fbsql_result (PHP 4 >= 4.0.6)

Get result data

mixed **fbsql_result** (resource result, int row [, mixed field])

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If

the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: fbsql_fetch_row(), fbsql_fetch_array(), and fbsql_fetch_object().

fbsql_rollback (PHP 4 >= 4.0.6)

Rollback a transaction to the database

bool **fbsql_rollback** ([resource link_identifier])

Returns: TRUE on success, FALSE on failure.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_commit()

fbsql_set_lob_mode (PHP 4 CVS only)

Set the LOB retrieve mode for a FrontBase result set

bool **fbsql_set_lob_mode** (resource result, string database_name)

Returns: TRUE on success, FALSE on error.

fbsql_set_lob_mode() sets the mode for retrieving LOB data from the database. When BLOB and CLOB data is stored in FrontBase it can be stored direct or indirect. Direct stored LOB data will always be fetched no matter the setting of the lob mode. If the LOB data is less than 512 bytes it will always be stored directly.

- FBSQL_LOB_DIRECT - LOB data is retrieved directly. When data is fetched from the database with fbsql_fetch_row(), and other fetch functions, all CLOB and BLOB columns will be returned as ordinary columns. This is the default value on a new FrontBase result.
- FBSQL_LOB_HANDLE - LOB data is retrieved as handles to the data. When data is fetched from the database with **fbsql_fetch_row ()**, and other fetch functions, LOB data will be returned as a handle to the data if the data is stored indirect or the data if it is stored direct. If a handle is returned it will be a 27 byte string formatted as "@'000000000000000000000000'".

See also: fbsql_create_blob(), fbsql_create_clob(), fbsql_read_blob(), and fbsql_read_clob().

fbsql_select_db (PHP 4 >= 4.0.6)

Select a FrontBase database

bool **fbsql_select_db** (string database_name [, resource link_identifier])

Returns: TRUE on success, FALSE on error.

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

The client contacts FBExec to obtain the port number to use for the connection to the database. If the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to fbsql_query() will be made on the active database.

if the database is protected with a database password, the user must call fbsql_database_password() before selecting the database.

See also: fbsql_connect(), fbsql_pconnect(), fbsql_database_password() and fbsql_query().

fbsql_start_db (PHP 4 >= 4.0.6)

Start a database on local or remote server

bool **fbsql_start_db** (string database_name [, resource link_identifier])

Returns: TRUE on success, FALSE on failure.

fbsql_start_db()

See also: fbsql_db_status() and fbsql_stop_db().

fbsql_stop_db (PHP 4 >= 4.0.6)

Stop a database on local or remote server

bool **fbsql_stop_db** (string database_name [, resource link_identifier])

Returns: TRUE on success, FALSE on failure.

fbsql_stop_db()

See also: fbsql_db_status() and fbsql_start_db().

fbsql_tablename (PHP 4 CVS only)

Get table name of field

string **fbsql_tablename** (resource result, int i)

fbsql_tablename() takes a result pointer returned by the fbsql_list_tables() function as well as an integer index and returns the name of a table. The fbsql_num_rows() function may be used to determine the number of tables in the result pointer.

Beispiel 1. fbsql_tablename() example

```
<?php
fbsql_connect ("localhost", "_SYSTEM", "");
$result = fbsql_list_tables ("wisconsin");
$i = 0;
while ($i < fbsql_num_rows ($result)) {
    $tb_names[$i] = fbsql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

fbsql_warnings (PHP 4 >= 4.0.6)

Enable or disable FrontBase warnings

bool **fbsql_warnings** ([bool OnOff])

Returns `TRUE` if warnings is turned on otherwise `FALSE`.

fbsql_warnings() enables or disables FrontBase warnings.

fbsql_database (PHP 4 >= 4.0.6)

Get or set the database name used with a connection

string **fbsql_database** (resource link_identifier [, string database])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

fbsql_get_autostart_info (PHP 4 >= 4.1.0)

No description given yet

array **fbsql_get_autostart_info** ([resource link_identifier])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

fbsql_hostname (PHP 4 >= 4.0.6)

Get or set the host name used with a connection

string **fbsql_hostname** (resource link_identifier [, string host_name])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

fbsql_password (PHP 4 >= 4.0.6)

Get or set the user password used with a connection

string **fbsql_password** (resource link_identifier [, string password])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

fbsql_set_transaction (PHP 4 CVS only)

Set the transaction locking and isolation

void **fbsql_set_transaction** (resource link_identifier, int Locking, int Isolation)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

fbsql_username (PHP 4 >= 4.0.6)

Get or set the host user used with a connection

string **fbsql_username** (resource link_identifier [, string username])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XXIX. filePro Funktionen

Diese Funktionen erlauben Lesezugriff auf die Daten, die in einer filePro Datenbank gespeichert sind.

filePro ist eingetragenes Warenzeichen von fP Technologies, Inc. Mehr Informationen zu filePro finden Sie unterhttp://www.fptech.com/.

filepro (PHP 3, PHP 4 >= 4.0.0)

lesen und überprüfen der Kartendatei

bool **filepro** (string directory)

Diese Funktion lies und überprüft die Kartendatei, in der die Feldanzahl und die Feldinformationen gespeichert sind.

Es wird keine Sperrung durchgeführt, daher sollten Sie eine Veränderung der filePro Datenbank, während sie in PHP geöffnet sein könnte, vermeiden.

filepro_fieldname (PHP 3, PHP 4 >= 4.0.0)

ermittelt des Namens eines Feldes

string **filepro_fieldname** (int field_number)

Gibt den Namen des Feldes, das *`field_number`* entspricht, zurück.

filepro_fieldtype (PHP 3, PHP 4 >= 4.0.0)

ermittelt den Typ eines Feldes

string **filepro_fieldtype** (int field_number)

Gibt den Typ des Feldes, das *`field_number`* entspricht, zurück.

filepro_fieldwidth (PHP 3, PHP 4 >= 4.0.0)

ermittelt die Breite eines Feldes

int **filepro_fieldwidth** (int field_number)

Gibt die Breite des Feldes, das *`field_number`* entspricht, zurück.

filepro_retrieve (PHP 3, PHP 4 >= 4.0.0)

empfängt Daten aus einer filePro Datenbank

string **filepro_retrieve** (int row_number, int field_number)

Gibt die Daten aus einem festgelegten Platz der Datenbank zurück.

filepro_fieldcount (PHP 3, PHP 4 >= 4.0.0)

bestimmt, wie viele Felder sich in einer filePro Datenbank befinden

int **filepro_fieldcount** (void)

Gibt die Anzahl der Felder (Spalten) in der geöffneten filePro Datenbank zurück.

Siehe auch filepro().

filepro_rowcount (PHP 3, PHP 4 >= 4.0.0)

bestimmt, wie viele Reihen sich in einer filePro Datenbank befinden

int **filepro_rowcount** (void)

Gibt die Anzahl der Reihen in der geöffneten filePro Datenbank zurück.

Siehe auch filepro().

XXX. Funktionen des Dateisystems

basename (PHP 3, PHP 4 >= 4.0.0)

Extrahiert den Namen einer Datei aus einer vollständigen Pfadangabe

string **basename** (string path [, string suffix])

Diese Funktion extrahiert aus einer vollständigen Pfadangabe den Namen der Datei und gibt diesen zurück. Endet der Dateiname mit *suffix*, wird dieses Ende ebenfalls abgeschnitten.

Unter Windows wird sowohl der Slash (/) als auch der Backslash (\) als Trennzeichen bei Pfadangaben benutzt. Unter anderen Betriebssystemen hingegen nur der Slash (/).

Beispiel 1. basename()

```
$path = "/home/httpd/html/index.php";
$file = basename($path);         // $file enthält den Wert "index.php"
$file = basename ($path,".php"); // $file enthält den Wert "index"
```

Anmerkung: Der Parameter *suffix* wurde in PHP 4.1.0. eingeführt.

Siehe auch dirname().

chgrp (PHP 3, PHP 4 >= 4.0.0)

Wechselt die Gruppenzugehörigkeit einer Datei

int **chgrp** (string filename, mixed group)

Weist der Datei *filename* die Benutzergruppe *group* zu (spezifiziert durch Name oder Nummer). Um diese Funktion ausführen zu können, müssen Sie Superuser sein. Andere Benutzer können die Gruppenzugehörigkeit einer Datei in jede Gruppe ändern, derer sie angehören.

Gibt bei Erfolg TRUE zurück, andernfalls FALSE.

Siehe auch chown() und chmod().

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

chmod (PHP 3, PHP 4 >= 4.0.0)

Ändert die Zugriffsrechte einer Datei

int **chmod** (string filename, int mode)

Diese Funktion ändert die Zugriffsrechte der Datei spezifiziert in *filename* in die Zugriffsrechte um, die in *mode* spezifiziert sind.

Bedenken Sie, das *mode* nicht automatisch für einen Oktalwert gehalten wird. Um sicher zu gehen, dass die Operation wie erwartet verläuft sollten Sie *mode* mit dem Prefix NULL (0) versehen:

```
chmod ("/somedir/somefile", 755);   // dezimal; möglicherweise falsch
chmod ("/somedir/somefile", "u+rwx,go+rx"); // String; falsch
chmod ("/somedir/somefile", 0755);  // oktal; richtiger Wert von mode
```

Gibt TRUE bei Erfolg zurück, andernfalls FALSE.

Siehe auch chown() und chgrp().

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

chown (PHP 3, PHP 4 >= 4.0.0)

Ändert den Eigentümer einer Datei

int **chown** (string filename, mixed user)

Ändert den Eigentümer der Datei *filename* in Benutzer *user*. Nur der Superuser kann den Eigentümer einer Datei ändern.

Gibt bei Erfolg TRUE zurück, andernfalls FALSE.

Siehe auch **chown()** und chmod().

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

clearstatcache (PHP 3, PHP 4 >= 4.0.0)

Löscht den Status Cache

void **clearstatcache** (void)

clearstatcache() löscht den Zwischenspeicher mit den Dateistatuswerten. Die Systemaufrufe stat oder lstat sind auf den meisten Systemen sehr "teure" Aufrufe. Daher wird das Ergebnis des letzten Aufrufs von einer der unten aufgeführten Funktionen für die nächste Benutzung eines ähnlichen Aufrufs mit dem gleichen Dateinamen gespeichert. Wenn Sie einen neuen Status-Check erzwingen wollen, z.B. wenn die Datei in der Zwischenzeit oftmals aufgerufen wurde, verändert oder verschwunden ist, können Sie diese Funktion dazu benutzen, das Ergebnis des letzten Aufrufes aus dem Speicher zu leeren.

Dieser Wert wird nur für die Zeit eines einzelnen Requests zwischengespeichert.

Folgende Funktionen betreffen den Dateistatus: stat(), lstat(), file_exists(), is_writeable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype(), and fileperms().

copy (PHP 3, PHP 4 >= 4.0.0)

Kopiert eine Datei

int **copy** (string source, string dest)

Die Funktion **copy()** kopiert eine Datei. Sie gibt `TRUE` zurück, wenn der Kopiervorgang erfolgreich war, andernfalls `FALSE`.

Beispiel 1. copy()

```
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...<br>\n");
}
```

Warnung

Sollte die Zieldatei bereits existieren, wird sie überschrieben.

Siehe auch move_uploaded_file(), rename(), und den Abschnitt über Steuerung von Dateiuploads.

delete (unknown)

"Dummy Handbuch Eintrag"

void **delete** (string file)

Dieses ist ein "Dummy Handbuch Eintrag" für diejenigen, die nach einer Funktion wie unlink() oder unset() an der falschen Stelle suchen.

Siehe auch unlink() um Dateien zu löschen, unset() um Variablen zu löschen.

dirname (PHP 3, PHP 4 >= 4.0.0)

Extrahiert den Verzeichnis-Namen aus einer vollständigen Pfadangabe

string **dirname** (string path)

Bei einer vollständigen Pfadangabe gibt diese Funktion den Verzeichnispfad zurück.

Unter Windows wird sowohl der Slash (/) als auch der Backslash (\) als Trennzeichen bei Pfadangaben benutzt. Unter anderen Betriebssystemen hingegen nur der Slash (/).

Beispiel 1. dirname()

```
$path = "/etc/passwd";
$file = dirname ($path); // $file enthält den Wert "/etc"
```

Anmerkung: Seit PHP 4.0.3 entspricht **dirname()** dem POSIX-Standard. Im Grunde heißt das, dass wenn keine Slashes in *path* vorhanden sind, wird ein Punkt ('.') zurückgegeben, womit das aktuelle Verzeichnis gemeint ist. Andernfalls enthält der zurückgegebene String *path*, jedoch ohne den hinteren /component. Beachten Sie, dass Sie von **dirname()** oft einen Slash oder Punkt zurückbekommen, wo die alte Funktionalität einen leeren String lieferte.

Siehe auch basename().

disk_free_space (PHP 4 >= 4.1.0)

Liefert den freien Speicherplatz in einem Verzeichnis

float **disk_free_space** (string directory)

Diese Funktion gibt den freien Speicherplatz eines Verzeichnisses in Byte zurück.

Beispiel 1. disk_free_space()

```
$df = disk_free_space("/"); // $df hält die Anzahl an freien Bytes
                            // im Verzeichnis  "/"
```

diskfreespace (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ist ein Alias für disk_free_space()

float **diskfreespace** (string directory)

Von diesem Alias wird abgeraten, benutzen Sie stattdessen disk_free_space().

disk_total_space (PHP 4 >= 4.1.0)

Liefert die Gesamtgröße eines Verzeichnisses

float **disk_total_space** (string directory)

Anhand des übergebenen Strings mit dem Verzeichnisnamen gibt diese Funktion die Anzahl Bytes auf dem korrespondierenden Dateisystem oder der Disk Partition zurück.

Beispiel 1. disk_total_space()

```
$df = disk_total_space("/"); // $df enthält die Gesamtanzahl
                             // verfügbarer Bytes in "/"
```

fclose (PHP 3, PHP 4 >= 4.0.0)

Schließt einen offenen Dateizeiger

bool **fclose** (int fp)

Die Datei, auf die der Dateizeiger *fp* zeigt wird geschlossen.

Gibt bei erfolgreichem Schließen der Datei `TRUE`, andernfalls `FALSE` zurück.

Der Datei-Zeiger muss gültig sein, d.h. die Datei, auf die gezeigt wird muss zuvor mit fopen() oder fsockopen() geöffnet worden sein.

feof (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob der Dateizeiger am Ende der Datei steht

int **feof** (int fp)

Gibt `TRUE` zurück, falls der Dateizeiger am Ende der Datei steht oder ein Fehler aufgetreten ist, andernfalls `FALSE`.

Der Dateizeiger muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit fopen(), popen() oder fsockopen() geöffnet worden sein.

fflush (PHP 4)

Schreibt den Ausgabepuffer in eine Datei

int **fflush** (int fp)

Diese Funktion erzwingt das Schreiben des gesamten Ausgabepuffers in die von dem Dateizeiger *fp* angegebenen Datei. Gibt bei Erfolg `TRUE`, andernfalls `FALSE` zurück.

Der Dateizeiger muss gültig sein, und auf eine mittels fopen(), popen(), oder fsockopen() erfolgreich geöffnete Datei zeigen.

fgetc (PHP 3, PHP 4 >= 4.0.0)

Liest das Zeichen, auf welches der Dateizeiger zeigt

string **fgetc** (int fp)

Gibt einen String zurück, der ein einzelnes Zeichen von der aktuellen Position des Dateizeigers *fp* enthält. Gibt `FALSE` zurück, wenn der Dateizeiger am Ende der Datei steht (wie die Funktion feof()).

Der Dateizeiger muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit fopen(), popen() oder fsockopen() geöffnet worden sein.

Siehe auch fread(), fopen(), popen(), fsockopen() und fgets().

fgetcsv (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Liest eine Zeile von der Position des Dateizeigers und prüft diese auf Komma-Separierte-Werte (CSV)

array **fgetcsv** (int fp, int length [, string delimiter])

Die Funktion **fgetcsv()** handelt ähnlich der Funktion fgets(), bis auf den Umstand, das **fgetcsv()** die eingelesene Zeile auf Kommaseparierte Felder hin überprüft. Das Ergebnis wird als Array zurückgegeben. Als Trennzeichen (Delimiter) wird - wenn kein anderes Trennzeichen im dritten optionalen Parameter angegeben wird - das Komma benutzt.

Der Dateizeiger *fp* muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit fopen(), popen() oder fsockopen() geöffnet worden sein.

'length' muss größer sein als die längste in der CSV-Datei vorhandene Zeile (um das Erkennen von EndOfLine(Ende der Zeile) Zeichen zu ermöglichen).

fgetcsv() gibt `FALSE` am Dateiende und bei Lesefehlern zurück.

Eine leere Zeile der CSV-Datei wird als Array mit einem Feld das den Wert `NULL` hält zurückgegeben. Sie wird nicht als Fehler behandelt.

Beispiel 1. fgetcsv() Beispiel - Lesen und Ausgabe des Inhalts einer CSV-Datei

```
$row = 1;                                   // Anzahl der Arrays
$fp = fopen ("test.csv","r");               // Datei wird zum Lesen
                                            // geöffnet
while ($data = fgetcsv ($fp, 1000, ",")) { // Daten werden aus der Datei
                                            // in einen Array $data gelesen
    $num = count ($data);                   // Felder im Array $data
                                            // werden gezählt
    print "<p> $num fields in line $row: <br>";
```

```
    $row++;                                 // Anzahl der Arrays wird
                                            // inkrementiert
    for ($c=0; $c < $num; $c++) {           // FOR-Schleife, um Felder
        print $data[$c] . "<br>";           // des Arrays auszugeben
    }
}
fclose ($fp);
```

fgets (PHP 3, PHP 4 >= 4.0.0)

Liest eine Zeile von der Position des Dateizeigers

string **fgets** (int fp, int length)

Gibt eine Zeile bis zu *length* -1 Bytes Länge zurück, welche aus der Datei von der aktuellen Position des Dateizeigers *fp* aus ausgelesen wird. Die Zeile endet an einem Zeilenumbruch (welcher im Rückgabewert enthalten ist), EOF (Ende der Datei) oder bei *length* - 1 Bytes (Je nachdem, was eher auftritt). Ist keine Länge angegeben, wird diese auf 1k bzw. 1024 Bytes gesetzt.

Wenn ein Fehler auftritt, wird `FALSE` zurückgegeben.

Häufig auftretende Fehler:

Wenn Sie an die C-Semantik von **fgets()** gewöhnt sind, sollten Sie auf die unterschiedliche Behandlung von `EOF` (genauer: wie EOF zurückgegeben wird) achten.

Der Datei-Zeiger *fp* muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit fopen(), popen(), oder fsockopen() geöffnet worden sein.

Ein einfaches Beispiel:

Beispiel 1. Eine Datei Zeile für Zeile einlesen

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
}
fclose ($fd);
```

Anmerkung: Der Parameter *length* wurde erst ab PHP 4.2.0 optional.

Siehe auch fread(), fopen(), popen(), fgetc(), fsockopen(), und socket_set_timeout().

fgetss (PHP 3, PHP 4 >= 4.0.0)

Liest eine Zeile von der Position des Dateizeigers und entfernt HTML und PHP-Tags.

string **fgetss** (int fp, int length [, string allowable_tags])

Diese Funktion ist identisch mit der Funktion fgets(), außer dass **fgetss()** versucht, vorhandene HTML und PHP-Tags aus dem gelesenen Text zu entfernen.

Optional können mit dem dritten Parameter `allowable_tags` Tags spezifiziert werden, die nicht entfernt werden sollen.

Anmerkung: `allowable_tags` wurden in PHP 3.0.13 bzw. PHP 4.0.0 hinzugefügt.

Siehe auch fgets(), fopen(), fsockopen(), popen() und strip_tags().

file (PHP 3, PHP 4 >= 4.0.0)

Liest eine Datei komplett in einen Array

array **file** (string filename [, int use_include_path])

Die Funktion **file()** ist identisch mit readfile(), außer dass die eingelesene Datei als Array zurückgegeben wird. Jedes Feld des Arrays korrespondiert mit einer Zeile der Datei. Der Zeilenumbruch bleibt erhalten.

Anmerkung: Jede Zeile in dem resultierenden Array enthält das Zeilenende ('\0'), weshalb Sie trim() verwenden müssen, falls Sie dieses Zeichen nicht vorhanden sein soll.

Optional kann der zweite Parameter benutzt und auf '1' gesetzt werden, wenn in dem include_path ebenfalls nach der Datei gesucht werden soll.

```
<?php
// liest eine Webseite in einen Array und gibt das Ergebnis aus
$fcontents = file ('http://www.php.net/');
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b>; ", htmlspecialchars ($line), "<br>\n";
}

// liefert eine Webseite in einen String
$fcontents = implode (", file ('http://www.php.net/'));
?>
```

Warnung

Diese Funktion ist (bis jetzt) nicht binary safe!

Tipp: Mit dieser Funktion können Sie eine URL als Dateinamen verwenden. falls Sie "fopen wrappers" ermöglicht haben. Für mehr Details siehe fopen().

Siehe auch readfile(), fopen(), fsockopen(), und popen().

file_exists (PHP 3, PHP 4 >= 4.0.0)

Überprüft, ob eine Datei existiert

bool **file_exists** (string filename)

Gibt `TRUE` zurück, wenn die mit *`filename`* spezifizierte Datei existiert, andernfalls `FALSE`.

file_exists() funktioniert nur mit lokalen Dateien. Die Datei, deren Existenz überprüft werden soll muss über das Dateisystem des Servers erreichbar sein.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

fileatime (PHP 3, PHP 4 >= 4.0.0)

Liefert Datum und Uhrzeit des letzten Zugriffs auf eine Datei

int **fileatime** (string filename)

Gibt Datum und Uhrzeit des letzten Zugriffs auf eine Datei zurück, oder `FALSE` wenn ein Fehler auftrat. Das Datum wird als UNIX-Timestamp zurückgegeben.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Anmerkung: Die atime einer Datei vorgesehen sich immer zu ändern, sobald Datenblöcke einer Datei geändert werden. Das kann sich folgenschwer auf die Performance auswirken, wenn eine Anwendung regelmäßig auf eine große Anzahl Dateien oder Verzeichnisse zugreift. Manche Unix Dateisysteme können mit deaktivierter atime-Aktualisierung gemountet werden, um die Performance solcher Applikationen zu erhöhen; USENET news spools sind ein häufiges Beispiel. Auf solchen Dateisystemen wird diese Funktion nutzlos.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

filectime (PHP 3, PHP 4 >= 4.0.0)

Liefert Datum und Uhrzeit der letzten Änderung des Dateizeigers Inode

int **filectime** (string filename)

Gibt Zeit der letzten Änderung des Dateizeigers Inode zurück, oder FALSE wenn ein Fehler auftrat. Die Zeit wird als UNIX-Timestamp zurückgegeben.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Anmerkung: Bei den meisten Unix Dateisystemen gilt eine Datei als geändert, sobald ihre Inode-Daten geändert wurden; d.h. wenn die Rechte, Eigentümer, Gruppe, oder andere Metadaten des Inodes aktualisiert werden. Siehe auch filemtime() (das ist was Sie suchen, wenn Sie Fußzeilen mit "Zuletzt Aktualisiert" für Webseiten erstellen möchten) und fileatime().

Beachten Sie auch, dass in einigen Unix-Texten ctime als die Zeit der Erstellung der Datei beschrieben wird. Das ist falsch. Bei den meisten Unix Dateisystemen gibt es keine Zeitangabe für die Erstellung von Unix Dateien.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

filegroup (PHP 3, PHP 4 >= 4.0.0)

Liefert die Gruppenzugehörigkeit einer Datei

int **filegroup** (string filename)

Gibt die Gruppen ID des Besitzers einer Datei zurück, oder FALSE wenn ein Fehler auftrat. Die Gruppen ID wird im numerischen Format zurückgegeben. Mit der Funktion posix_getgrgid() können Sie diese zu ihrem Gruppennamen auflösen.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

Anmerkung: Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

fileinode (PHP 3, PHP 4 >= 4.0.0)

Liefert die Inode-Nummer einer Datei

int **fileinode** (string filename)

Gibt die Inode-Nummer einer Datei zurück, oder FALSE wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

filemtime (PHP 3, PHP 4 >= 4.0.0)

Liefert Datum und Uhrzeit der letzten Dateiänderung

int **filemtime** (string filename)

Gibt Datum und Uhrzeit zurück, wann die Datei das letzte mal modifiziert wurde, oder FALSE wenn ein Fehler auftrat. Das Datum wird als UNIX-Timestamp zurückgegeben.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Anmerkung: Diese Funktion gibt den Zeitpunkt zurück, an dem in die Datenblöcke einer Datei geschrieben wurde; d.h. den Zeitpunkt der Änderung des Inhalts der Datei. Verwenden Sie date(), um das Ergebnis dieser Funktion in ein ausgabefähiges Änderungsdatum für z.B. Fußzeilen in Webseiten zu erhalten.

fileowner (PHP 3, PHP 4 >= 4.0.0)

Liefert den Dateieigentümer

int **fileowner** (string filename)

Gibt die Benutzer ID des Eigentümers einer Datei zurück, oder FALSE wenn ein Fehler auftrat. Die Benutzer ID wird in numerischem Format zurückgegeben und kann mit der Funktion posix_getpwuid() zum Benutzernamen aufgelöst werden.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

fileperms (PHP 3, PHP 4 >= 4.0.0)

Liefert die Zugriffsrechte einer Datei

int **fileperms** (string filename)

Gibt die Zugriffsrechte einer Datei zurück, oder FALSE wenn ein Fehler auftrat.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

filesize (PHP 3, PHP 4 >= 4.0.0)

Liefert die Größe einer Datei

int **filesize** (string filename)

Gibt die Größe einer Datei in Bytes zurück, oder FALSE wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

filetype (PHP 3, PHP 4 >= 4.0.0)

Liefert den Typ einer Datei

string **filetype** (string filename)

Gibt den Typ einer Datei zurück. Mögliche Werte sind fifo, char, dir, block, link, file und unknown.

Gibt FALSE zurück, wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

flock (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Portables Datei-Verriegelungs-Verfahren

bool **flock** (int fp, int operation [, int wouldblock])

PHP liefert einen portablen Weg, Dateien in einer "beratenden" Art und Weise zu verriegeln. D.h. alle zugreifende Programme müssen die gleiche Art und Weise der Verriegelung benutzen, oder es funktioniert nicht.

flock() arbeitet mit `fp` welcher ein existierender Dateizeiger sein muss. Der Parameter `operation` bestimmt die Art der Verriegelung und kann einen der folgenden Werte annehmen:

- Um eine verteilte, nur lesende Verriegelung zu erhalten, setzten Sie `operation` auf LOCK_SH (bzw. 1 vor PHP 4.0.1).
- Um eine exklusive, nur schreibende Verriegelung zu erhalten, setzen Sie `operation` auf LOCK_EX (bzw. 2 vor PHP 4.0.1).
- Um eine Verriegelung freizugeben (lesend oder schreibend), setzten Sie `operation` auf LOCK_UN (bzw. 3 vor PHP 4.0.1).
- Wenn Sie Zugriffe während der Verriegelung durch **flock()** erlauben wollen, fügen Sie LOCK_NB (bzw. 4 vor PHP 4.0.1) zu `operation` hinzu.

flock() erlaubt es Ihnen ein einfaches Leser/Schreiber Modell zu erstellen, welches auf fast jeder Plattform benutzt werden kann (inkl. den meisten UNIX-Derivaten und auch Windows). Das optionale dritte Argument ist auf `TRUE` gesetzt, wenn die Verriegelung sperren sollte (EWOULDBLOCK errno Bedingung).

flock() gibt `TRUE` bei Erfolg, `FALSE` wenn ein Fehler auftrat (z.B. wenn eine Verriegelung nicht erstellt werden konnte) zurück.

> **Anmerkung:** Da **flock()** einen Dateizeiger benötigt, könnten Sie eine spezielle Sperrdatei benutzen, um den Zugriff auf eine Datei zu schützen, welche Sie mittels Öffnen im Schreibmodus (mit dem Argument "w" or "w+" von fopen()) kürzen wollen.

> **Warnung**
>
> **flock()** funktioniert nicht auf NFS und verschiedenen anderen vernetzten Dateisystemen. Für weitere Detail konsultieren Sie bitte Ihre Betriebssystemdokumentation.
>
> Bei einigen Betriebssystemen ist **flock()** auf dem Prozesslevel implementiert. Wenn Sie ein multithreaded Server API wie ISAPI benutzen können Sie sich nicht auf **flock()** verlassen, um Dateien vor anderen PHP-Skripten zu schützen, welche in parallelen Threads der gleichen Server-Instanz laufen!

fopen (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine Datei oder URL

int **fopen** (string filename, string mode [, int use_include_path])

Wenn `filename` mit "http://" (nicht case sensitive) beginnt, wird zu dem spezifiziertem Server eine HTTP 1.0 Connection geöffnet, die Seite mittels der HTTP GET Methode angefordert, und ein

Dateizeiger auf den Anfang des Textes der Antwort zurückgegeben. Ein 'Host:' header wird mit der Anfrage geschickt, um namens-basierte Virtual Hosts zu verarbeiten.

Beachten Sie, dass Ihnen der Dateizeiger nur den Empfang des *body* aus der Antwort erlaubt; mit dieser Funktion können Sie auf den HTTP response header nicht zugreifen. Weiters sind HTTP Verbindungen nur zum Lesen.

In älteren Versionen als PHP 4.0.5 werden Weiterleitungen (HTTP Redirects) nicht unterstützt, hier muss die Datei daher vollständig und mit abschließendem "/" beschrieben werden.

Wenn `filename` mit "ftp://" (nicht case sensitive) beginnt, wird eine ftp-Connection zu dem spezifizierten Server geöffnet und ein Zeiger auf die angeforderte Datei zurückgegeben. Wenn der Server keinen passiven FTP-Modus unterstützt, wird diese Methode fehlschlagen. Sie können Dateien entweder zum Schreiben oder zum Lesen öffnen, jedoch nicht für beides gleichzeitig.

Wenn `filename` "php://stdin", "php://stdout", oder "php://stderr" ist, wird der korrespondierende stdio stream geöffnet (Feature wurde in Version 3.0.13 eingeführt, in früheren Versionen muss der Dateiname wie z.B. "/dev/stdin" oder "/dev/fd/0" benutzt werden, um auf die stdio streams zuzugreifen).

Wenn `filename` mit irgendetwas anderem beginnt, wird die Datei vom lokalen Dateisystem geöffnet und ein Dateizeiger auf die geöffnete Datei zurückgegeben.

Wenn das Öffnen scheitert, gibt die Funktion `FALSE` zurück.

`mode` kann einen der folgenden Werte einnehmen:

- 'r' - öffnet die Datei nur zum Lesen und positioniert den Dateizeiger auf den Anfang der Datei.
- 'r+' - Öffnet die Datei zum Lesen und Schreiben und setzt den Dateizeiger auf den Anfang der Datei.
- 'w' - Öffnet die Datei nur zum Schreiben und setzt den Dateizeiger auf den Anfang der Datei sowie die Länge der Datei auf 0 Byte. Wenn die Datei nicht existiert wird versucht sie anzulegen.
- 'w+' - Öffnet die Datei zum Lesen und Schreiben und setzt den Dateizeiger auf den Anfang der Datei sowie die Länge der Datei auf 0 Byte. Wenn die Datei nicht existiert, wird versucht sie anzulegen.
- 'a' - Öffnet die Datei nur zum Schreiben. Positioniert den Dateizeiger auf das Ende der Datei. Wenn die Datei nicht existiert, wird versucht sie anzulegen.
- 'a+' - Öffnet die Datei zum Lesen und Schreiben. Positioniert den Dateizeiger auf das Ende der Datei. Wenn die Datei nicht existiert, wird versucht sie anzulegen.

Zusätzlich kann `mode` der Buchstabe 'b' hinzugefügt werden, der die Behandlung von Binär-Dateien erlaubt. Dies ist nur auf Systemen sinnvoll, welche zwischen Binär- und Text-Dateien unterscheiden (z.B. Windows. Ist bei Unix sinnlos). Wenn das Feature nicht gebraucht wird, wird es einfach ignoriert.

Sie können optional ein dritten Parameter benutzen und diesen auf '1' setzten, wenn Sie auch im include_path nach der Datei suchen möchten.

Beispiel 1. fopen()

```
$fp = fopen ("/home/simi/simi.txt", "r");
$fp = fopen ("/home/simi/simi.gif", "wb");
$fp = fopen ("http://www.php.net/", "r");
$fp = fopen ("ftp://user:password@example.com/", "w");
```

Wenn Sie Probleme mit dem Lesen oder Schreiben von Dateien haben und PHP als Servermodul benutzen, stellen Sie zunächst sicher, dass die Dateien und Verzeichnisse die Sie benutzen wollen auch für den Server-Prozess zugänglich sind (Rechtevergabe).

Achten Sie auf Windows-Systemen darauf, dass Sie als Verzeichnistrenner normale Schrägstriche '/' benutzen um plattformunabhängig programmieren zu können. Sollten Sie dennoch Backslashes verwenden, vergessen Sie nicht diese zu escapen '\\'

```
$fp = fopen ("c:\\data\\info.txt", "r");
```

Siehe auch fclose(), fsockopen(), socket_set_timeout(), und popen().

fpassthru (PHP 3, PHP 4 >= 4.0.0)

Gibt alle verbleibenden Daten eines Dateizeigers direkt aus.

int **fpassthru** (int fp)

Die Funktion liest alle Daten von der aktuellen Position des Dateizeigers bis EOF aus und gibt die Daten auf der Standartausgabe (Browser) aus.

Wenn ein Fehler auftritt, gibt **fpassthru()** FALSE zurück.

Der Dateizeiger muss gültig sein und auf eine erfolgreich durch fopen(), popen() oder fsockopen() geöffnete Datei zeigen. Vielleicht müssen Sie rewind() aufrufen um den Dateizeiger auf den Beginn der Datei zurückzusetzen, wenn Sie bereits Daten in die Datei geschrieben haben. Die Datei wird geschlossen, wenn **fpassthru()** die Datei komplett gelesen hat (hinterlässt nutzlosen *fp*-Dateizeiger).

Wenn Sie den Inhalt einer Datei nur auf der Standartausgabe (Browser) ausgeben wollen, sollten Sie besser readfile() benutzen, bei der Sie nicht zuvor die Datei mittels fopen() geöffnet haben müssen.

> **Anmerkung:** Wenn Sie **fpassthru()** an einer Binärdatei unter Windows benutzen sollten Sie sich vergewissern, dass Sie die Datei im Binärmodus öffnen, indem Sie ein `b` an den Mode-Parameter von fopen() anhängen.

Siehe auch readfile(), fopen(), popen() und fsockopen()

fputs (PHP 3, PHP 4 >= 4.0.0)

Schreibt Daten an die Position des Dateizeigers

int **fputs** (int fp, string str [, int length])

fputs() ist ein Alias für die Funktion fwrite(), und ist in jeder Hinsicht identisch mit ihr. Beachten Sie, dass der Parameter *length* optional ist, und dass, wenn er nicht benutzt wird, die gesamte Zeichenkette geschrieben wird.

fread (PHP 3, PHP 4 >= 4.0.0)

Liest Binärdaten aus einer Datei

string **fread** (int fp, int length)

fread() liest *length* Bytes vom Dateizeiger, maximal jedoch bis zum Dateiende.

```
// liest den Inhalt einer Datei in einen String
$filename = "/usr/local/simi.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Anmerkung: Auf Systemen, welche zwischen Binär- und Textdateien (z.B. Windows) unterscheiden, muss die Datei mit 'b' im fopen() Mode-Parameter geöffnet werden.

```
$filename = "c:\\files\\somepic.gif";
$fd = fopen ($filename, "rb");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Siehe auch fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), fscanf(), file() und fpassthru().

fscanf (PHP 4)

Interpretiert den Input einer Datei entsprechend einem angegebenen Format

mixed **fscanf** (int handle, string format [, string var1])

Die Funktion **fscanf()** ist sscanf() ähnlich, wobei sie ihren Input aus der mit *handle* angegebenen Datei nimmt, und entsprechend dem angegebenen *format* interpretiert. Sind nur 2 Parameter angegeben, werden die analysierten Werte als Array zurückgegeben. Werden auch optionale

Parameter übergeben, gibt die Funktion die Anzahl der ermittelten Werte zurück. Die optionalen Parameter müssen referenziert übergeben werden.

Beispiel 1. fscanf()

```
$fp = fopen ("users.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... Tue etwas mit den Werten
}
fclose($fp);
```

Beispiel 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
robert  slacker us
luigi   florist it
```

Siehe auch fread(), fgets(), fgetss(), sscanf(), printf(), und sprintf().

fseek (PHP 3, PHP 4 >= 4.0.0)

Positioniert den Dateizeiger

int **fseek** (int fp, int offset [, int whence])

Setzt den Dateizeiger einer mit *fp* angegebenen Datei auf eine neue Position. Diese neue Position, angegeben in Bytes vom Begin der Datei an, wird definiert durch Hinzufügen von *offset* zu der Position spezifiziert durch *whence*, deren Werte wie folgt definiert sind:
SEEK_SET - Setzt Position gleich *offset* bytes.
SEEK_CUR - Setzt Position auf die aktuelle Stelle plus *offset*.
SEEK_END - Setzt die Position ans Ende der Datei plus *offset*. (Um zu einer Position vor EOF zu gelangen, überge

Wenn *whence* nicht angegeben wurde, wird SEEK_SET angenommen.

Bei Erfolg wird '0' zurückgegeben, andernfalls '-1'. Beachten Sie, dass die Überschreitung des Dateiendes (EOF) nicht als Fehler bewertet wird.

Diese Funktion sollte nicht auf Dateizeiger angewandt werden, die durch die Verwendung von "http://" oder "ftp://" bei der Funktion fopen() erhalten wurden.

Anmerkung: Das *whence*-Argument wurde nach PHP 4.0.0 eingeführt.

Siehe auch ftell() und rewind().

fstat (PHP 4 >= 4.0.0)

Liefert Informationen über eine Datei mit offenem Dateizeiger

array **fstat** (int fp)

Trägt die Statistiken der mittel des Dateizeigers fp geöffneten Datei zusammen. Diese Funktion ist ähnlich der Funktion stat(), außer dass sie mit einem offenen Dateizeiger anstatt eines Dateinamens arbeitet.

Gibt die Statistiken einer Datei in Form eines Arrays mit den folgenden Elementen zurück:

1. Gerät
2. Inode
3. Anzahl Links
4. User id des Eigentümers
5. Gruppen id Eigentümer
6. Gerätetype wenn ein Inode-Gerät *
7. Größe in Bytes
8. Zeit des letzten Zugriffs
9. Zeit der letzten Modifikation
10. Zeit der letzten Änderung
11. Blockgröße für Dateisystem I/O *
12. Anzahl der Blöcke

* - nur gültig bei Systemen, welche den st_blksize Typ unterstützen, andere Systeme (z.B. Windows) geben -1 zurück

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe clearstatcache() für weitere Einzelheiten.

ftell (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die aktuelle Position des Dateizeigers

int **ftell** (int fp)

Gibt die aktuelle Position des Dateizeigers `fp` zurück, z.B. seinen Offset im Dateistream.

Wenn ein Fehler auftritt, wird `FALSE` zurückgegeben.

Der Dateizeiger muss gültig sein und auf eine erfolgreich durch fopen() oder popen() geöffnete Datei zeigen.

Siehe auch fopen(), popen(), fseek() und rewind().

ftruncate (PHP 4 >= 4.0.0)

Kürzt eine Datei auf die angegebene Länge

int **ftruncate** (int fp, int size)

Nimmt den Dateizeiger `fp`, und verkürzt die Datei auf die Länge `size`. Diese Funktion gibt bei Erfolg `TRUE`, ansonsten `FALSE` zurück.

fwrite (PHP 3, PHP 4 >= 4.0.0)

Schreibt Binärdaten in eine Datei

int **fwrite** (int fp, string string [, int length])

fwrite() schreibt den Inhalt einer Zeichenkette `string` in die Datei, auf welche der `fp` Dateizeiger zeigt. Wenn der `length` Parameter gegeben ist, wird das Schreiben nach `length` Bytes beendet, oder wenn das Dateiende (EOF) erreicht ist, je nachdem, was eher eintritt.

fwrite() gibt bei Erfolg die Anzahl der geschriebenen Bytes zurück, andernfalls -1.

Bedenken Sie, dass wenn der `length` Parameter gesetzt ist, die magic_quotes_runtime Konfigurations-Option ignoriert wird und somit keine Slashes mehr von der Zeichenkette `string` entfernt werden.

> **Anmerkung:** Auf Systemen, welche zwischen Binär- und Textdateien (z.B. Windows) unterscheiden, muss die Datei mit 'b' im fopen() Mode-Parameter geöffnet werden.

Siehe auch fread(), fopen(), fsockopen(), popen() und fputs().

set_file_buffer (PHP 3>= 3.0.8, PHP 4)

Setzt die Dateipufferung für einen gegebenen Dateizeiger

int **set_file_buffer** (int fp, int buffer)

Eine Ausgabe mittels fwrite() wird normalerweise mit 8k gepuffert. Das heißt, dass wenn zwei Prozesse in den gleichen Output Stream (eine Datei) schreiben wollen, jeder nach 8k Daten unterbrochen wird, um den anderen schreiben zu lassen. **set_file_buffer()** setzt die Pufferung für Schreiboperationen des gegebenen Dateizeigers `fp` auf `buffer` Bytes. Wenn `buffer` 0 ist, erfolgen die Schreiboperationen ungepuffert. Dies stellt sicher, dass alle Schreiboperationen mit fwrite() abgeschlossen werden, bevor andere Prozesse in diesen Output Stream schreiben können.

Die Funktion gibt bei Erfolg 0, oder EOF wenn die Anfrage nicht erfolgreich war, zurück.

Das folgende Beispiel zeigt, wie man mittels **set_file_buffer()** einen ungepufferten Stream generiert.

Beispiel 1. set_file_buffer()

```
$fp=fopen($file, "w");
if($fp){
  set_file_buffer($fp, 0);
  fputs($fp, $output);
  fclose($fp);
}
```

Siehe auch fopen(), fwrite().

is_dir (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob der gegebene Dateiname ein Verzeichnis ist

bool **is_dir** (string filename)

Gibt `TRUE` zurück, wenn der Dateiname existiert und es sich hierbei um ein Verzeichnis handelt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

Siehe auch is_file() und is_link().

is_executable (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Datei ausführbar ist

bool **is_executable** (string filename)

Gibt `TRUE` zurück, wenn die Datei existiert und ausführbar ist.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Siehe auch is_file() und is_link().

is_file (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob der Dateiname eine reguläre Datei ist

bool **is_file** (string filename)

Gibt `TRUE` zurück, wenn die Datei existiert und eine reguläre Datei ist.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

Siehe auch is_dir() und is_link().

is_link (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob der Dateiname ein symbolischer Link ist

bool **is_link** (string filename)

Gibt TRUE zurück, wenn der Dateiname existiert und es sich um einen symbolischen Link handelt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

Siehe auch is_dir(), is_file() und readlink().

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

is_readable (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Datei lesbar ist

bool **is_readable** (string filename)

Gibt TRUE zurück wenn die Datei existiert und sie lesbar ist.

Bedenken Sie, dass PHP mit der Benutzer ID auf die Datei zugreift, unter welcher der Webserver läuft (oftmals ist dieses 'nobody'). Beschränkungen durch safe_mode werden nicht berücksichtigt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Siehe auch is_writable().

is_writable (PHP 4 >= 4.0.0)

Prüft, ob in eine Datei geschrieben werden kann

bool **is_writable** (string filename)

Gibt `TRUE` zurück, wenn die Datei existiert und nicht schreibgeschützt ist. Das Dateinamen-Argument *`filename`* kann ein Verzeichnisname sein. So können Sie überprüfen, ob in ein Verzeichnis geschrieben werden kann.

Bedenken Sie, dass PHP mit der Benutzer ID auf die Datei zugreift, unter welcher der Webserver läuft (oftmals ist dieses 'nobody'). Beschränkungen durch safe_mode werden nicht berücksichtigt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

Diese Funktion ist nicht für remote Dateien geeignet, die zu prüfende Datei muss über das Dateisystem des Servers verfügbar sein.

Siehe auch is_readable().

is_writeable (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob in eine Datei geschrieben werden kann

bool **is_writeable** (string filename)

Dies ist ein Alias für is_writable()

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Prüft, ob die Datei mittels HTTP POST upgeloaded wurde.

bool **is_uploaded_file** (string filename)

Gibt `TRUE` zurück, wenn die Datei `filename` mittels HTTP POST upgeloaded wurde. Das ist hilfreich sicherzustellen, dass kein böswilliger User versucht hat das Script zu überlisten, indem er es Dateien bearbeiten lässt, die es gar nicht bearbeiten sollte. Zum Beispiel `/etc/passwd`.

Diese Prüfung ist speziell dort interessant, wo irgendeine Aktion mit upgeloadeten Dateien deren Inhalt dem User zugänglich machen könnte (oder auch anderen Usern auf dem selben System).

is_uploaded_file() ist in PHP 3 nur nach Version 3.0.16, und in PHP 4 nur nach Version 4.0.2 verfügbar. Sollten Sie eine frühere Version benutzen, können Sie sich mit der folgenden Funktion weiterhelfen:

> **Anmerkung:** Dies funktioniert *nicht* in Versionen nach PHP 4.0.2. Das liegt an der internen Funktionalität von PHP, welche sich nach dieser Version geändert hat.

```
<?php
/* Userland Test für hochgeladene Datei. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam(", "));
    }
    $tmp_file .= '/' . basename($filename);
    /* Der Benutzer könnte einen führenden Slash in php.ini haben... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
```

```
}

/* So verwenden Sie sie, nachdem Sie in diesen älteren
 * Versionen auch keine move_uploaded_file() haben: */
if (is_uploaded_file($HTTP_POST_FILES['userfile'])) {
    copy($HTTP_POST_FILES['userfile'], "/place/to/put/uploaded/file");
} else {
    echo "Mögliche Dateiupload-Attacke: Dateiname '$HTTP_POST_FILES[userfile]'.";
}
?>
```

Siehe auch move_uploaded_file(), und das Kapitel Steuerung von Dateiuploads für ein einfaches Anwendungsbeispiel.

link (PHP 3, PHP 4 >= 4.0.0)

Erzeugt einen absoluten Link

int **link** (string target, string link)

link() erzeugt einen absoluten Link mit dem Ziel `target`.

Siehe auch symlink() um symbolische Links zu erstellen und readlink() zusammen mit linkinfo().

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

linkinfo (PHP 3, PHP 4 >= 4.0.0)

Liefert Informationen über einen Link

int **linkinfo** (string path)

linkinfo() gibt das st_dev Feld des UNIX C stat structure resultierend aus dem lstat System Call zurück. Diese Funktion wird benutzt, um zu verifizieren ob ein Link (auf den mit `path` gezeigt wird) wirklich existiert (Hierbei wird die gleiche Methode benutzt, die in dem S_ISLNK Makro in stat.h definiert ist). Gibt 0 oder `FALSE` zurück wenn ein Fehler auftrat.

Siehe auch symlink(), link(), und readlink().

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

mkdir (PHP 3, PHP 4 >= 4.0.0)

Erstellt ein Verzeichnis

int **mkdir** (string pathname, int mode)

Versucht ein Verzeichnis mit dem spezifiziertem Namen zu erstellen.

Bedenken Sie, dass sie wahrscheinlich den Mode als oktalen Wert spezifizieren wollen, was bedeutet, dass er eine voranstehende Null haben sollte. Der Mode wird ebenfalls mit der aktuellen umask modifiziert, welchen Sie mittels umask() ändern können.

```
mkdir ("/path/to/my/dir", 0700);
```

Gibt `TRUE` bei Erfolg, andernfalls `FALSE` zurück.

Siehe auch rmdir().

move_uploaded_file (PHP 4 >= 4.0.3)

Verschiebt eine upgeloadete Datei an einen neuen Ort.

bool **move_uploaded_file** (string filename, string destination)

Diese Funktion prüft, ob die mit *`filename`* bezeichnete Datei eine gültige Upload-Datei ist (d.h., dass sie mittels PHP's HTTP POST Upload-Mechanismus upgeloaded wurde). Ist die Datei gültig, wird sie zum in *`destination`* bezeichneten Dateinamen verschoben.

Wenn *`filename`* keine gültige Datei ist, wird keine Aktion ausgeführt, und **move_uploaded_file()** gibt `FALSE` zurück.

Wenn *`filename`* eine gültige Datei ist, jedoch aus irgendeinem Grund nicht verschoben werden kann, wird keine Aktion ausgeführt, und **move_uploaded_file()** gibt `FALSE` zurück. Zusätzlich wird eine Warnung ausgegeben.

Diese Prüfung ist speziell dort interessant, wo irgendeine Aktion mit upgeloadeten Dateien deren Inhalt dem User zugänglich machen könnte (oder auch anderen Usern auf dem selben System).

Anmerkung: move_uploaded_file() ist von den normalen safe-mode UID-Einschränkungen nicht betroffen. Dies ist nicht unsicher, da **move_uploaded_file()** nur mit via PHP hochgeladenen Dateien arbeitet.

Warnung

Sollte die Zieldatei bereits existieren, wird sie überschrieben.

Siehe auch is_uploaded_file(), und das Kapitel Steuerung von Dateiuploads für ein einfaches Anwendungsbeispiel.

parse_ini_file (PHP 4 >= 4.0.0)

Analysiert eine Konfiguratiionsdatei

array **parse_ini_file** (string filename [, bool process_sections])

parse_ini_file() lädt die in *filename* angegebene Datei, und gibt die darin enthaltenen Einstellungen in einem assoziativen Array zurück. Setzen Sie den letzten Parameter *process_sections* auf TRUE, erhalten Sie ein mehrdimensionales Array mit den Gruppennamen und Einstellungen. Ist *process_sections* nicht angegeben, wird FALSE angenommen.

> **Anmerkung:** Diese Funktion hat nichts mit der php.ini zu tun, denn diese ist bereits abgearbeitet, wenn Sie Ihr Skript ausführen. Diese Funktion ist vorgesehen, um Konfigurationsdateien für Ihre eigenen Applikationen einzulesen.

Die Struktur der ini Datei ist der von php.ini ähnlich.

Beispiel 1. Inhalt der sample.ini

```
; Dies ist ein Beispiel für eine Konfigurationsdatei
; Kommentare beginnen wie in der php.ini mit ';'

[erste_gruppe]
eins = 1
fünf = 5

[zweite_gruppe]
pfad = /usr/local/bin
```

Beispiel 2. parse_ini_file()

```
<?php

// Ohne Gruppen analysieren
$ini_array = parse_ini_file("sample.ini");
print_r($ini_array);

// Mit Gruppen analysieren
$ini_array = parse_ini_file("sample.ini", TRUE);
print_r($ini_array);

?>
```

Würde wie folgt ausgeben:

```
Array
(
    [eins] => 1
    [fünf] => 5
    [pfad] => /usr/local/bin
)
Array
(
    [erste_gruppe] => Array
        (
            [eins] => 1
            [fünf] => 5
        )

    [zweite_gruppe] => Array
        (
            [pfad] => /usr/local/bin
        )

)
```

pathinfo (PHP 4 >= 4.0.3)

Liefert Informationen über den Dateipfad

array **pathinfo** (string path)

pathinfo() gibt ein assoziatives Array mit Informationen über den Pfad *path* zurück. Die folgenden Arrayelemente werden zurückgegeben: den Verzeichnisnamen *dirname*, den Dateinamen *basename*, und die Dateinamenserweiterung *extension*.

Beispiel 1. pathinfo()

```
<?php

$path_parts = pathinfo("/www/htdocs/index.html");

echo $path_parts["dirname"] . "\n";
echo $path_parts["basename"] . "\n";
echo $path_parts["extension"] . "\n";

?>
```

Würde ausgeben:

```
/www/htdocs
index.html
html
```

Siehe auch dirname(), basename(), parse_url() und realpath().

pclose (PHP 3, PHP 4 >= 4.0.0)

Schließt einen Prozess-Dateizeiger

int **pclose** (int fp)

Schließt einen Prozess-Dateizeiger, der durch einen Aufruf von popen() geöffnet wurde.

Der Dateizeiger muss gültig sein und durch einen erfolgreichen Funktionsaufruf von popen() zurückgegeben worden sein.

Gibt den End-Status eines laufenden Prozesses zurück.

Siehe auch popen().

popen (PHP 3, PHP 4 >= 4.0.0)

Öffnet einen Prozesszeiger

int **popen** (string command, string mode)

Öffnet eine Verbindung zu einem Prozess, der durch die Anweisung *command* ausgeführt wurde.

Gibt einen Dateizeiger zurück, der identisch ist mit dem, der bei der Funktion fopen() zurückgegeben wird, außer dass dieser nur in eine Richtung funktioniert (entweder lesend oder schreibend) und mit der Funktion pclose() geschlossen werden muss. Dieser Zeiger kann mit den Funktionen fgets(), fgetss() und fputs() benutzt werden.

Wenn ein Fehler auftritt, gibt diese Funktion `FALSE` zurück.

```
$fp = popen ("/bin/ls", "r");
```

Anmerkung: Wenn das auszuführende Kommando nicht gefunden werden konnte, wird eine gültige Ressource zurückgegeben. Das mag seltsam erscheinen, macht aber Sinn. Es ermöglicht Ihnen, auf von der Shell zurückgegebene Fehlermeldungen zuzugreifen:

```
<?php
error_reporting(E_ALL);

/* Umleitung hinzufügen, dass wir stderr erhalten können */
$fp = popen('/path/to/spooge 2>&1', 'r');
echo "'$fp'; " . gettype($fp) . "\n";
$read = fread($fp, 2096);
echo $read;
pclose($fp);
?>
```

Siehe auch pclose().

readfile (PHP 3, PHP 4 >= 4.0.0)

Gibt eine Datei aus

int **readfile** (string filename [, int use_include_path])

Liest den Inhalt einer Datei und gibt ihn an der Standartausgabe (Browser) aus.

Gibt die Anzahl der Bytes einer Datei zurück. Wenn ein Fehler auftritt wird `FALSE` zurückgegeben und eine Fehlermeldung wird auf der Standartausgabe ausgegeben. Die Ausgabe einer Fehlermeldung kann durch Voranstellen eines '@' verhindert werden (@readfile).

Wenn *`filename`* mit "http://" (nicht case sensitive) beginnt, wird eine HTTP 1.0 Verbindung zu dem spezifiziertem Server hergestellt und der Text der Antwort wird auf die Standartausgabe geschrieben.

In älteren Versionen als PHP 4.0.5 werden Weiterleitungen (HTTP Redirects) nicht unterstützt, hier muss die Datei daher vollständig und mit abschließendem "/" beschrieben werden.

Wenn *`filename`* mit "ftp://" (nicht case sensitive) beginnt, wird eine FTP Verindung zu dem spezifiziertem Server hergestellt und die angeforderte Datei wird auf der Standartausgabe ausgegeben. Wenn der Server keinen passiven Mode unterstützt, wird diese Funktion fehlschlagen.

Wenn *`filename`* mit keinem dieser Zeichenketten beginnt, wird die Datei vom lokalen Dateisystem geöffnet und ihr Inhalt auf der Standartausgabe ausgegeben.

Sie können optional den zweiten Parameter benutzen und diesen auf '1' setzen, wenn Sie auch im include_path nach der Datei suchen möchten.

Siehe auch fpassthru(), file(), fopen(), include(), require() und virtual().

readlink (PHP 3, PHP 4 >= 4.0.0)

Liefert das Ziel eines symbolischen Links

string **readlink** (string path)

readlink() ist von der Funktionalität her identisch mit der 'readlink C Funktion' und gibt den Inhalt des Pfades des symbolischen Links zurück oder 0, wenn der Aufruf fehlschlug.

Siehe auch is_link(), symlink() und linkinfo().

> **Anmerkung:** Diese Funktion ist auf Windows-Plattformen nicht implementiert.

rename (PHP 3, PHP 4 >= 4.0.0)

Benennt eine Datei um

int **rename** (string oldname, string newname)

Versucht eine Datei `oldname` in `newname` umzubenennen.

Gibt `TRUE` bei Erfolg, andernfalls `FALSE` zurück.

rewind (PHP 3, PHP 4 >= 4.0.0)

Setzt den Dateizeiger auf das erste Byte der Datei

int **rewind** (int fp)

Setzt den Dateizeiger (Datei-Positions-Indikator) für `fp` an den Anfang des Dateistreams.

Wenn ein Fehler auftritt, gibt die Funktion 0 zurück.

Der Dateizeiger muss gültig sein und auf eine Datei zeigen, die zuvor erfolgreich durch fopen() geöffnet wurde.

> **Anmerkung:** Haben Sie eine Datei im Append-Modus ("a") geöffnet, werden unabhängig von der Dateiposition alle zu schreibenden Daten angehängt.

Siehe auch fseek() und ftell().

rmdir (PHP 3, PHP 4 >= 4.0.0)

Löscht ein Verzeichnis

int **rmdir** (string dirname)

Versucht ein Verzeichnis spezifiziert durch die Pfadangabe zu löschen. Das Verzeichnis muss leer sein und die relevanten Rechte müssen erteilt sein.

Wenn ein Fehler auftritt wird 0 zurückgegeben.

Siehe auch mkdir().

stat (PHP 3, PHP 4 >= 4.0.0)

Liefert diverse Informationen über eine Datei

array **stat** (string filename)

Sammelt Informationen über eine Datei spezifiziert in `filename`.

Gibt einen Array mit folgenden statistischen Informationen als Array- Elemente zurück:

1. Laufwerk
2. inode
3. inode protection mode
4. Anzahl der Links
5. Benutzer ID des Inhabers
6. Gruppen ID des Inhabers
7. Laufwerkstyp wenn Inode-Laufwerk *
8. Größe in Bytes
9. Zeitpunkt des letzten Zugriffs
10. Zeitpunkt der letzten Modifizierung
11. Zeitpunkt der letzten Änderung
12. Blockgröße für das Dateisystem I/O *
13. Anzahl der zugewiesenen Blöcke

* - nur gültig auf Systemen, die st_blksize unterstützen. Andere Systeme (z.B. Windows) geben '-1' zurück.

Gibt FALSE zurück, wenn ein Fehler auftrat.

stat() ist nicht für remote Dateien geeignet.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

lstat (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Liefert Informationen über eine Datei oder einen symbolischen Link.

array **lstat** (string filename)

Ermittelt statistische Informationen über eine Datei oder symbolischen Link, spezifiziert in `filename`. Diese Funktion ist identisch mit der Funktion stat(), außer dass wenn der Parameter `filename` ein symbolischer Link ist, dessen Status zurückgegeben wird und nicht der Status der Datei, auf die der symbolische Link zeigt.

Gibt einen Array mit statistischen Informationen über die Datei als Array-Elemente zurück:

1. Laufwerk
2. inode

3. Anzahl der Links
4. Benutzer ID des Inhabers
5. Gruppen ID des Inhabers
6. Laufwerkstyp wenn inode Laufwerk *
7. Größe in Bytes
8. Zeitpunkt des letzten Zugriffs
9. Zeitpunkt der letzten Modifikation
10. Zeitpunkt der letzen Änderung
11. Blockgröße für das Dateisystem I/O *
12. Anzahl der zugewiesenen Blöcke

* - nur gültig auf Systemen, die st_blksize unterstützen. Andere Systeme (z.B. Windows) geben '-1' zurück.

lstat() ist nicht für remote Dateien geeignet.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch clearstatcache() für weitere Einzelheiten.

realpath (PHP 4 >= 4.0.0)

Erzeugt einen kanonisch absoluten Pfadnamen

string **realpath** (string path)

realpath() expandiert alle symbolischen Links, und beseitigt Verweise zu '/./', '/../' und extra '/' Zeichen im Input *path*, und gibt den absoluten Pfadnamen kanonisch zurück. Der Pfad im Ergebnis enthält keine Komponenten eines symbolischen Links mehr, wie '/./' or '/../'.

Beispiel 1. realpath()

```
$real_path = realpath("../../index.php");
```

symlink (PHP 3, PHP 4 >= 4.0.0)

Erzeugt einen symbolischen Link

int **symlink** (string target, string link)

symlink() erzeugt einen symbolischen Link von dem gegebenen Parameter *target* mit dem spezifiziertem Namen *link*.

Siehe auch link() um absolute Links zu erstellen, und readlink() zusammen mit linkinfo().

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

tempnam (PHP 3, PHP 4 >= 4.0.0)

Erzeugt eine Datei mit eindeutigem Dateinamen

string **tempnam** (string dir, string prefix)

Erzeugt eine Datei mit einem eindeutigen Dateinamen in dem spezifizierten Verzeichnis. Wenn das Verzeichnis nicht existiert, erstellt **tempnam()** eine Datei im temporären Verzeichnis des Systems, und gibt den Dateinamen zurück.

Vor PHP 4.0.6 war das Verhalten von **tempnam()** systemabhängig. Unter Windows wird die TMP-Umgebungsvariable den *dir* Parameter überschreiben, unter Linux-Systemen hat die TMPDIR Umgebungsvariable Vorrang, während SVR4 immer den spezifizierten Verzeichnisnamen benutzen wird, wenn dieses existiert. Wenn Sie nicht genau wissen, wie Ihr System **tempnam()** behandelt, dann konsultieren Sie bitte Ihre System-Dokumentation.

Gibt den neuen temporären Dateinamen bei Erfolg, oder eine `FALSE` Zeichenkette wenn ein Fehler auftritt zurück.

Beispiel 1. tempnam()

```
$tmpfname = tempnam ("/tmp", "FOO");

$fp = fopen($tmpfname, "w");
fwrite($fp, "schreibe in Tämporärdatei");
fclose($fp);

// tue hier etwas

unlink($tmpfname);
```

Anmerkung: Diese Funktion hat ihr Verhalten seit PHP 4.0.3 geändert. Die temporäre Datei wird ebenfalls angelegt um eine "Hetzsituation" zu vermeiden, wo in der Zeit nach dem Generieren des Strings, und bevor das Script dazukommt die Datei anzulegen, die Datei im Dateisystem erscheinen könnte. Beachten Sie, dass falls Sie die Datei nicht mehr benötigen, Sie diese wieder entfernen müssen (dies geschieht nicht automatisch).

Siehe auch tmpfile() und unlink().

tmpfile (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Legt eine temporäre Datei an

int **tmpfile** (void)

Erstellt eine temporäre Datei mit einem eindeutigen Dateinamen im Schreibmodus und gibt einen Dateizeiger ähnlich wie es bei der Funktion fopen() erfolgt, zurück. Die Datei wird automatisch gelöscht, wenn sie geschlossen wird (mit fclose()), oder das Skript beendet wird.

Für Details konsultieren Sie bitte die Dokumentation Ihres Betriebssystems bzgl. der `tmpfile(3)` Funktion, wie auch die `stdio.h` Header-Datei.

Beispiel 1. tmpfile()

```
$temp = tmpfile();
fwrite($temp, "schreibt in Temporärdatei");
fclose($temp); // dies entfernt die Datei
```

Siehe auch tempnam().

touch (PHP 3, PHP 4 >= 4.0.0)

Setzt die Zugriffs- und Modifizierungszeit einer Datei

int **touch** (string filename [, int time])

Setzt Datum und Uhrzeit der letzten Änderung und des letzten Zugriffs der durch *`filename`* spezifizierten Datei auf den durch *`time`* spezifizierten Wert. Wenn die Option *`time`* nicht angegeben wurde, wird die aktuelle Zeit benutzt. Dies ist äquivalent zu dem, was utime tut (manchmal wird auch als utimes davon gesprochen).

Wenn die Datei nicht existiert, wird sie erzeugt.

Gibt `TRUE` bei Erfolg, andernfalls `FALSE` zurück.

Beispiel 1. touch()

```
if (touch ($DateiName)) {
    print "$DateiName Modifizierungs-Zeit wurde auf das aktuelle
          Datum und die aktuelle Uhrzeit gesetzt.";
} else {
    print "Tut mir leid, Modifikations-Zeit von $DateiName konnte
          nicht geändert werden.";
}
```

umask (PHP 3, PHP 4 >= 4.0.0)

Ändert die aktuelle umask (Zugriffsrechte)

int **umask** (int mask)

umask() setzt PHP's umask auf *mask* & 0777 und gibt die alte umask zurück. Wenn PHP als Servermodul benutzt wird, wird die umask zurückgespeichert wenn die jeweilige Anfrage beendet ist.

umask() ohne Argumente gibt einfach die aktuelle umask zurück.

Anmerkung: Diese Funktion ist auf Windows-Plattformen nicht implementiert.

unlink (PHP 3, PHP 4 >= 4.0.0)

Löscht eine Datei

int **unlink** (string filename)

Löscht *filename*. Ähnlich der UNIX C unlink() Funktion.

Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

Siehe auch rmdir() um Verzeichnisse zu löschen.

XXXI. Forms Data Format Funktionen

Forms Data Format (FDF) ist ein Format zur Verarbeitung von Formularen innerhalb von PDF Dokumenten. Weiterführende Informationen zu FDF und dessen Verwendung sind unter http://partners.adobe.com/asn/developer/acrosdk/forms.html zu finden.

> **Anmerkung:** Bei Problemen mit der Konfiguration der FdfTk-Unterstützung sollte man überprüfen, ob die Datei FdfTk.h sowie die Bibliothek libFdfTk.so im richtigen Verzeichnis zu finden sind. Die korrekten Verzeichnisse sind fdftk-dir/include und fdftk-dir/lib. Das ausschließliche Entpacken der Distribution führt nicht zu dem gewünschten Ergebnis.

Der Ansatz den FDF verfolgt, ist vergleichbar mit dem üblicher HTML Formulare. Die wesentlichen Unterschiede liegen in der Übertragung der Daten nach Absenden des Formulars (dies ist das eigentliche Form Data Format), sowie in dem Format des Dokuments welches das Formular enthält(PDF). Die Verarbeitung der FDF Daten ist eine der Möglichkeiten, die die FDF Funktionen bereitstellen. Darüberhinaus kann man beispielsweise innerhalb eines vorhandenen PDF Formulars die Input Felder dynamisch mit Daten füllen, ohne das Formular selbst zu verändern. Hierzu erzeugt man das FDF Dokument (fdf_create()), setzt die Werte der jeweiligen Input Felder (fdf_set_value()) und verbindet es mit dem jeweiligen PDF Formular (fdf_set_file()). Nachdem man das FDF Dokument, mit dem passenden MimeType `application/vnd.fdf` versehen, ausgegeben hat, erkennt das Acrobat Reader PlugIn des Browsers den MimeType und füllt das Zielformular mit den Daten des FDF Dokuments.

Betrachtet man ein FDF Dokument mit einem Text Editor, findet sich dort ein Katalog Objekt `FDF`. Dieses Objekt kann mehrere Einträge der Art `Fields`, `F`, `Status` etc.. enthalten. Die gebräuchlichsten Einträge sind `Fields`, welches zu einer Anzahl von Input Feldern verweist, sowie `F` das den Dateinamen des zugehörigen PDF Dokuments enthält. Diese Einträge finden sich in der FDF Dokumentation unter /F-Key oder /Status-Key. Modifikationen dieser Einträge werden mit Hilfe der Funktionen fdf_set_file() und fdf_set_status() vorgenommen. Felder werden u.a. mit fdf_set_value(), fdf_set_opt() modifiziert.

Das Beispiel zeigt die Auswertung der Formular Daten.

Beispiel 1. Auswertung eines FDF Dokuments

```
<?php
// Speichern der FDF Daten in eine temporäre Datei
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Öffnen der temporären Datei und Auswertung der Daten
// Das PDF Formular enthält verschiedene Eingabefelder mit den zugehörigen Namen
// band, datum, kommentar, verleger, bearbeiter, sowie zwei Checkboxen
// zeige_verleger und zeige_bearbeiter.

$fdf = fdf_open("test.fdf");
$band = fdf_get_value($fdf, "band");
echo "Das Feld Band hat den Wert '<B>$band</B>'<BR>";

$datum = fdf_get_value($fdf, "datum");
echo "Das Feld Datum hat den Wert '<B>$datum</B>'<BR>";
```

```
$kommentar = fdf_get_value($fdf, "kommentar");
echo "Das Feld Kommentar hat den Wert '<B>$kommentar</B>'<BR>";

if (fdf_get_value($fdf, "zeige_verleger") == "On") {
    $verleger = fdf_get_value($fdf, "verleger");
    echo "Das Feld Verleger hat den Wert '<B>$verleger</B>'<BR>";
} else {
  echo "Verleger soll nicht angezeigt werden.<BR>";
}

if (fdf_get_value($fdf, "zeige_bearbeiter") == "On") {
  $bearbeiter = fdf_get_value($fdf, "bearbeiter");
  echo "Das Feld Bearbeiter hat den Wert '&<B>$bearbeiter</B>'<BR>";
} else {
  echo "Bearbeiter soll nicht angezeigt werden.<BR>";
}
fdf_close($fdf);
?>
```

fdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Öffnet ein FDF Dokument

int **fdf_open** (string filename)

Die Funktion **fdf_open()** öffnet eine Datei mit Formulardaten. Diese Datei muß die von einem PDF Formular zurückgegebenen Formulardaten enthalten. Momentan muss diese Datei 'manuell' erzeugt werden. Mit Hilfe von fopen() schreibt man den Inhalt von HTTP_FDF_DATA mit fwrite() in die Datei. Eine automatische Vorbelegung von Variablen mit den Daten der Eingabefelder, wie sie bei HTML Formularen möglich ist, existiert in diesem Fall nicht.

Beispiel 1. Zugang zu den Fornmulardaten

```
<?php
// Speichern der FDF Daten in eine temporäre Datei
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Öffnen der temporären Datei und Auswertung der Daten
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

Siehe auch fdf_close().

fdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schließt ein FDF Dokument

bool **fdf_close** (int fdf_document)

Die Funktion **fdf_close()** schließt das FDF Dokument.

Siehe auch fdf_open().

fdf_create (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Erzeugt ein neues FDF Dokument

int **fdf_create** (void)

Die Funktion **fdf_create()** erzeugt ein neues FDF Dokument. Diese Funktion wird benötigt, wenn man Eingabefelder in einem PDF Dokument mit Daten belegen möchte.

Beispiel 1. Füllen von Formularwerten in einem PDF Dokument

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "band", $band, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

Siehe auch fdf_close(), fdf_save(), fdf_open().

fdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Speichern eines FDF Dokuments

int **fdf_save** (string filename)

Die Funktion **fdf_save()** speichert ein FDF Dokument. Das FDF Toolkit stellt eine Möglichkeit zur Verfügung, das Dokument an der Standardausgabe auszugeben, wenn der Parameter *`filename`* '.' ist. Dies funktioniert allerdings nicht, wenn PHP als Apache Module verwendet wird. In diesem Fall sollte man die Ausgabe in eine Datei schreiben und die Funktion fpassthru() verwenden, um den Inhalt anzuzeigen.

Siehe auch fdf_close() and example for fdf_create().

fdf_get_value (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den Wert eines Feldes zurück

string **fdf_get_value** (int fdf_document, string fieldname)

Die Funktion **fdf_get_value()** gibt den Wert eines Feldes zurück.

Siehe auch fdf_set_value().

fdf_set_value (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Wert eines Feldes

bool **fdf_set_value** (int fdf_document, string fieldname, string value, int isName)

Die Funktion **fdf_set_value()** setzt den Wert eines Feldes. Der letzte Parameter bestimmt, ob der Wert des Feldes in einen PDF Namen (`isName` = 1) konvertiert oder als PDF String (`isName` = 0) behandelt wird.

Siehe auch fdf_get_value().

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den Namen des nächsten Feldes zurück

string **fdf_next_field_name** (int fdf_document, string fieldname)

Die Funktion **fdf_next_field_name()** gibt den Namen des Feldes nach `fieldname` oder den Namen des ersten Feldes zurück, wenn der Parameter `NULL` ist.

Siehe auch **fdf_set_field()**, **fdf_get_field()**.

fdf_set_ap (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Legt das Aussehen eines Feldes fest

bool **fdf_set_ap** (int fdf_document, string field_name, int face, string filename, int page_number)

Die Funktion **fdf_set_ap()** legt das Aussehen eines Feldes (d.h den Wert des /AP Schlüssels) fest. Mögliche Werte von `face` sind 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_status (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Wert des /STATUS Schlüssels

bool **fdf_set_status** (int fdf_document, string status)

Die Funktion **fdf_set_status()** setzt den Wert des /STATUS Schlüssels.

Siehe auch fdf_get_status().

fdf_get_status (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den Wert des /STATUS Schlüssels zurück

string **fdf_get_status** (int fdf_document)

Die Funktion **fdf_get_status()** gibt den Wert des /STATUS Schlüssels zurück.

Siehe auch fdf_set_status().

fdf_set_file (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Wert des /F Schlüssels

bool **fdf_set_file** (int fdf_document, string filename)

Die Funktion **fdf_set_file()** setzt den Wert des /F Schlüssels. Der /F Schlüssel ist eine Referenz zu einem PDF Formular welches mit Daten gefüllt werden soll. In einer Web Umgebung ist diese Referenz ein URL (z.B. http:/testfdf/resultlabel.pdf).

Siehe auch fdf_get_file() und das Beispiel zu fdf_create().

fdf_get_file (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den Wert des /F Schlüssels zurück

string **fdf_get_file** (int fdf_document)

Die Funktion fdf_set_file() gibt den Wert des /F Schlüssels zurück.

Siehe auch fdf_set_file().

fdf_set_flags (PHP 4 >= 4.0.2)

Setzt Flags eines Feldes

bool **fdf_set_flags** (int fdf_document, string fieldname, int whichFlags, int newFlags)

Die Funktion **fdf_set_flags()** setzt bestimmte Flags des Feldes *fieldname*.

Siehe auch fdf_set_opt().

fdf_set_opt (PHP 4 >= 4.0.2)

Setzt die Option eines Feldes

bool **fdf_set_opt** (int fdf_document, string fieldname, int element, string str1, string str2)

Die Funktion **fdf_set_opt()** setzt Optionen des Feldes *fieldname*.

Siehe auch fdf_set_flags().

fdf_set_submit_form_action (PHP 4 >= 4.0.2)

Setzt die 'Submit Form Action' für ein Feld

bool **fdf_set_submit_form_action** (int fdf_document, string fieldname, int trigger, string script, int flags)

Die Funktion **fdf_set_submit_form_action()** setzt die 'Submit Form Action' für das Feld *fieldname*.

Siehe auch fdf_set_javascript_action().

fdf_set_javascript_action (PHP 4 >= 4.0.2)

Weist einem Feld eine Javascript Aktion zu

bool **fdf_set_javascript_action** (int fdf_document, string fieldname, int trigger, string script)

fdf_set_javascript_action() weist dem Feld *fieldname* eine Javascript Aktion zu.

Siehe auch fdf_set_submit_form_action().

fdf_set_encoding (PHP 4 >= 4.1.0)

Legt die Kodierung des FDF Zeichensatzes fest

bool **fdf_set_encoding** (int fdf_document, string encoding)

fdf_set_encoding() legt die Kodierung des Zeichensatzes innerhalb des FDF Dokuments *fdf_document* fest. In Acrobat 5.0 sind "Shift_JIS", "UHC", "GBK","BigFive" gültige Werte für den Parameter *encoding*.

Die Funktion **fdf_set_encoding()** ist ab PHP-4.1.0 oder später verfügbar.

fdf_add_template (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Fügt dem FDF Dokument ein Template hinzu

bool **fdf_add_template** (int fdfdoc, int newpage, string filename, string template, int rename)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XXXII. FriBiDi functions

fribidi_log2vis (PHP 4 >= 4.0.4)

Einen logischen String in einen visuellen konvertieren

string **fribidi_log2vis** (string str, string direction, int charset)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XXXIII. FTP-Funktionen

FTP ist die Abkürzung für File Transfer Protocol (Datei-Übertragungs-Protokoll).

Die folgenden Konstanten sind definiert, sobald das FTP-Modul benutzt wird: `FTP_ASCII` und `FTP_BINARY`.

Beispiel 1. ftp()-Beispiel:

```
<?php
// Herstellen der Basis-Verbindung
$conn_id = ftp_connect("$ftp_server");

// Einloggen mit Benutzername und Kennwort
$login_result = ftp_login($conn_id, "$ftp_user_name", "$ftp_user_pass");

// Verbindung überprüfen
if ((!$conn_id) || (!$login_result)) {
        echo "Ftp-Verbindung nicht hergestellt!";
        echo "Verbindung mit $ftp_server als Benutzer $user nicht möglich";
        die;
    } else {
        echo "Verbunden mit $ftp_server als Benutzer $user";
    }

// Upload der Datei
$upload = ftp_put($conn_id, "$destination_file", "$source_file", FTP_BINARY);

// Upload-Status überprüfen
if (!$upload) {
        echo "Ftp upload war fehlerhaft!";
    } else {
        echo "Datei $source_file auf $ftp_server als $destination_file geschrieben";
    }

// Schließen des FTP-Streams
ftp_quit($conn_id);
?>
```

ftp_connect (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Stellt eine FTP-Verbindung her

int **ftp_connect** (string Host [, int Port])

Gibt bei Erfolg einen Stream zurück, sonst `FALSE`.

ftp_connect() stellt eine FTP-Verbindung zum angegebenen *`Host`* her. Der *`Port`*-Parameter enthält optional einen alternativer Port für die Verbindung. Fehlt er oder ist er `NULL`, wird der Standard-FTP-Port 21 benutzt.

ftp_login (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Anmelden einer FTP-Verbindung (Login)

int **ftp_login** (int ftp_stream, string Benutzername, string Passwort)

Wenn die Anmeldung erfolgreich war, wird `TRUE`, anderenfalls `FALSE` zurück gegeben.

Anmeldung beim angegebenen FTP-Stream.

ftp_pwd (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Gibt den aktuellen Verzeichnis-Namen zurück

string **ftp_pwd** (int ftp_stream)

Gibt das aktuelle Verzeichnis oder - im Fehlerfall - `FALSE` zurück.

ftp_cdup (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Wechselt in das um eine Ebene höhere Verzeichnis

int **ftp_cdup** (int ftp_stream)

Bei Erfolg wird `TRUE`, ansonsten `FALSE` zurück gegeben.

Geht eine Verzeichnisebene höher.

ftp_chdir (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Verzeichnis-Wechsel auf einem FTP-Server

int **ftp_chdir** (int ftp_stream, string Verzeichnis)

Bei Erfolg wird `TRUE`, ansonsten `FALSE` zurück gegeben.

Wechselt in den durch *Verzeichnis* angegebenen Datei-Ordner.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Erzeugt ein Verzeichnis

string **ftp_mkdir** (int ftp_stream, string Verzeichnis)

Bei Erfolg wird der Name des erzeugten Ordners, ansonsten FALSE zurück gegeben.

Erzeugt das angegebene *Verzeichnis*.

ftp_rmdir (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Löscht ein Verzeichnis

int **ftp_rmdir** (int ftp_stream, string Verzeichnis)

Bei Erfolg wird TRUE, ansonsten FALSE zurück gegeben.

Löscht das angegebene *Verzeichnis*.

ftp_nlist (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Gibt eine Liste der im angegebenen Verzeichnis enthaltenen Dateien zurück

array **ftp_nlist** (int ftp_stream, string Verzeichnis)

Gibt ein Array aus Dateinamen oder - bei Fehler - FALSE zurück.

ftp_rawlist (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Gibt eine detaillierte Liste der Dateien in einem angegebenen Verzeichnis zurück

int **ftp_rawlist** (int ftp_stream, string Verzeichnis)

ftp_rawlist() führt den FTP LIST-Befehl aus und gibt das Ergebnis in einem Array zurück. Jedes Array-Element entspricht einer Textzeile. Die Ausgabe wird von PHP in keinster Weise ausgewertet. Das System-Art-Merkmal, welches per ftp_systype()-Funktion zurück gegeben wird, kann benutzt werden um die Auswertungsart der Ergebnisrückgabe zu ermitteln.

ftp_systype (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ermittelt den Systemtyp des entfernten FTP-Servers

string **ftp_systype** (int ftp_stream)

Gibt den entfernten Systemtyp an bzw. `FALSE` bei einem Fehler.

ftp_pasv (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Schaltet den passiven Modus ein oder aus

int **ftp_pasv** (int ftp_stream, int pasv)

Bei Erfolg wird `TRUE`, ansonsten `FALSE` zurück gegeben.

ftp_pasv() schaltet in den passiven Modus, wenn der *`pasv`*-Parameter `TRUE` ist (ausschalten per *`pasv`*-Parameter gleich `FALSE`). Im passiven Modus werden Datenverbindungen durch den Client statt durch den Server initiiert.

ftp_get (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liest eine Datei des FTP-Servers und speichert sie lokal (download)

int **ftp_get** (int ftp_stream, string local_file, string remote_file, int mode)

Bei Erfolg wird `TRUE`, sonst `FALSE` zurück gegeben.

ftp_get() lädt das *`remote_file`* vom FTP-Server und speichert es unter *`local_file`* auf ihren lokalen Arbeitsplatz. Der anzugebende Transfer-Parameter *`mode`* muss entweder `FTP_ASCII` oder `FTP_BINARY` sein.

ftp_fget (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Lädt eine Datei vom FTP-Server und speichert sie in eine geöffnete, lokale Datei (download)

int **ftp_fget** (int ftp_stream, int fp, string remote_file, int mode)

Sofern erfolgreich wird `TRUE`, ansonsten `FALSE` zurück gegeben.

ftp_fget() lädt das *`remote_file`* vom FTP-Server und schreibt es in die per Datei-Pointer angegebene Datei *`fp`*. Der Transfer-Modus-Parameter *`mode`* muss entweder FTP_ASCII oder FTP_BINARY sein.

ftp_put (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Überträgt eine Datei auf einen FTP-Server (upload)

int **ftp_put** (int ftp_stream, string remote_file, string local_file, int mode)

Bei Erfolg wird `TRUE`, sonst `FALSE` zurück gegeben.

ftp_put() speichert die per *local_file* bestimmte Datei auf den FTP-Server als Datei namens *remote_file*. Der Übertragungs-Modus-Parameter *mode* muss entweder FTP_ASCII oder FTP_BINARY lauten.

Beispiel 1. ftp_put()-Beispiel:

```
$upload = ftp_put ($conn_id, "$destination_file", "$source_file", FTP_ASCII);
```

ftp_fput (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Übertragt eine geöffnete Datei auf einen FTP-Server (upload)

int **ftp_fput** (int ftp_stream, string remote_file, int fp, int mode)

Bei Erfolg wird TRUE, sonst FALSE zurück gegeben.

ftp_fput() übertragt das per Datei-Zeiger *fp* benannte und geöffnete File zum FTP-Server. Die übertragenen Daten werden in der über *remote_file* bestimmten Datei auf dem FTP-Server gespeichert. Der *mode*-Übertragungs-Parameter muss entweder FTP_ASCII oder FTP_BINARY sein.

ftp_size (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ermittelt die Dateigrösse einer angegebenen Datei

int **ftp_size** (int ftp_stream, string remote_file)

Gibt die Dateigrösse oder -1 bei Fehlern zurück.

ftp_size() ermittelt die Grösse einer Datei. Wenn ein Fehler vorliegt, so z.B. wenn die Datei nicht existiert, wird -1 zurück gegeben. Nicht alle FTP-Server unterstützen diese Funktionalität.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ermittelt die letzte Änderungszeit der angegebenen Datei

int **ftp_mdtm** (int ftp_stream, string remote_file)

Bei Erfolg wird ein UNIX-Timestamp zurück gegeben, sonst -1.

ftp_mdtm() prüft den Zeitpunkt der letzten Dateiänderung und liefert diesen als UNIX-Timestamp zurück. Beim Auftreten eines Fehlers wird -1 zurück gegeben. Nicht alle FTP-Server unterstützen diese Funktionalität.

Anmerkung: ftp_mdtm() funktioniert nicht mit Verzeichnissen.

ftp_rename (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Benennt eine Datei auf dem FTP-Server um

int **ftp_rename** (int ftp_stream, string from, string to)

Falls erfolgreich wird `TRUE`, sonst `FALSE` zurück gegeben.

ftp_rename() benennt die mit *`from`* angegebene Datei in *`to`*um.

ftp_delete (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Löscht eine Datei auf dem FTP-Server

int **ftp_delete** (int ftp_stream, string path)

Falls erfolgreich wird `TRUE`, sonst `FALSE` zurück gegeben.

ftp_delete() löscht die per *`path`* angegebene Datei auf dem FTP-Server.

ftp_site (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Sendet ein SITE-Kommando zum Server

int **ftp_site** (int ftp_stream, string cmd)

Bei erfolgreicher Ausführung wird `TRUE`, sonst `FALSE` zurück gegeben

ftp_site() sendet das in *`cmd`* stehende Kommando zum FTP-Server. SITE-Kommandos sind nicht standardisiert und von Server zu Server verschieden. Sie sind nützlich, um Sachen wie Datei-Zugriffsrechte und Gruppen-Mitgliedschaften zu handhaben.

ftp_quit (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Schließt / beendet eine FTP-Verbindung

int **ftp_quit** (int ftp_stream)

ftp_connect() schließt die per *`ftp_stream`* bezeichnete FTP-Sitzung.

XXXIV. Function Handling functions

These functions all handle various operations involved in working with functions.

call_user_func (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Aufruf einer benutzerdefinierten Funktion

mixed **call_user_func** (string function_name [, mixed parameter [, mixed ...]])

Ruft die Funktion mit Namen *function_name* mit den Parametern *parameters* auf.

```
function friseur ($type) {
    print "Sie möchten einen $type-Kopf, kein Problem<br>";
}
call_user_func ('friseur', "Bubi");
call_user_func ('friseur', "Pilz");
```

create_function (PHP 4)

Erzeugen einer anonymen / temporären (Lambda-Stil) Funktion

string **create_function** (string args, string code)

Erzeugt eine anonyme Funktion an Hand der übergebenen Parameter und gibt einen eindeutigen Namen für diese zurück. Normalerweise sind die *args* als mit "'" begrenzte Strings zu übergeben. Dies gilt auch für *code*. Dadurch werden die Variablen vor dem Parsen geschützt. Falls sie """ verwenden, müssen sie die Variablen-Namen "escapen", wie z.B. `\$avar`.

Dieser Befehl kann z.B. verwendet werden, um eine Funktion an Hand von Informationen zu erzeugen, die erst zur Laufzeit vorliegen.

Beispiel 1. Erzeugen einer anonymen / temporären Funktion per create_function():

```
$newfunc = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "Neue Temp-Funktion: $newfunc\n";
echo $newfunc(2,M_E)."\n";
// Ausgabe:
// Neue Temp-Funktion: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Sie können auch eine allgemeine Handhabungs-Funktion erstellen, die einen mehr oder weniger große Anzahl an Operationen mit einer Liste von Parametern durchführt.

Beispiel 2. Beispiel zum o.a. Verwendungszweck:

```
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
        echo $farr[$f]($var1,$var2)."\n";
}

// Erzeugung eines Bündels an mathematischen Funktionen
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
```

```
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b;} else {return false;}';
$farr = array(
    create_function('$x,$y', 'return "als trig: ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "als Hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );

echo "\nGebrauch des ersten Arrays der anonymen Funktionen\n";
echo "Parameter: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// Erzeugung eines Bündels an String-Behandlungs-Routinen
$garr = array(
    create_function('$b, $a','if (strncmp ($a, $b, 3) == 0) return "** \"$a\" '.
        'und \"$b\"\n** Sieht für mich gleich aus! (ansehen der ersten 3 Zeichen)";'),
    create_function('$a, $b','; return "CRCs: ".crc32($a)." , ".crc32(b);'),
    create_function('$a, $b','; return "ähnlich(a, b) = ".similar_text($a, $b, &$p)."($p%
    );
echo "\nGebrauch des zweiten Arrays der anonymen Funktionen\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
```

wenn sie nun den o.a Code ausführen lassen, wird folgendes ausgegeben :

```
Gebrauch des ersten Arrays der anonymen Funktionen
Parameter: 2.3445, M_PI
als trig: -1.6291725057799
als Hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Gebrauch des zweiten Arrays der anonymen Funktionen
** "Twas the night" und "Twas brilling and the slithy toves"
** Sieht für mich gleich aus! (ansehen der ersten 3 Zeichen)
CRCs: -725381282 , 1908338681
ähnlich(a,b) = 11(45.833333333333%)
```

Möglicherweise liegt der häufigste Anwendungsfall für Funktionen im Lambda-Stil (anonym) bei den Callback-Funktionen, also z.B. beim Gebrauch von array_walk() oder usort().

Beispiel 3. Gebrauch der anonymen Funktionen als Callback-Funktion:

```
$av = array("der ","ein ","dieser ","jener ");
array_walk($av, create_function('&$v,$k','$v = $v."Zug";'));
print_r($av);  // mit PHP3 benutzen sie bitte var_dump()
// Ausgaben:
// Array
// (
//     [0] => der Zug
//     [1] => ein Zug
//     [2] => dieser Zug
```

```
//     [3] => jener Zug
// )

// Ein Array aus Strings nach Länge geordnet
$sv = array("kurz","länger","noch länger","dieser ist am längsten");
print_r($sv);
// Ausgaben:
// Array
// (
//     [0] => kurz
//     [1] => länger
//     [2] => noch länger
//     [3] => dieser ist am längsten
// )

// Sortierung von lang nach kurz
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// Ausgabe:
// Array
// (
//     [0] => dieser ist am längsten
//     [1] => noch länger
//     [2] => länger
//     [3] => kurz
// )
```

func_get_arg (PHP 4 >= 4.0.0)

Gibt einen Eintrag aus einer Liste von Argumenten zurück

int **func_get_arg** (int arg_num)

Gibt das Argument des an der *arg_num* stehenden Stelle in eine benutzer-definierte Liste zurück. Der Zähler hierfür beginnt bei 0. Der Aufruf von **func_get_arg()** ausserhalb einer Funktions-Definition erzeugt eine Warnung.

Ist *arg_num* grösser als die Anzahl der vorhandenen Argumente, wird eine Warnung aus- und FALSE zurück gegeben.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Anzahl der Argumente: $numargs<br>\n";
    if ($numargs >= 2) {
        echo "Das 2. Argument ist: " . func_get_arg (1) . "<br>\n";
    }
}

foo (1, 2, 3);
?>
```

func_get_arg() kann in Verbindung mit func_num_args() und func_get_args() gebraucht werden, um benutzer-definierte Funktionen mit variabler Argument-Liste zu erstellen.

Anmerkung: *Diese Funktion steht ab PHP 4 zur Verfügung.*

func_get_args (PHP 4 >= 4.0.0)

Gibt ein Array zurück, das die Liste der einer Funktion übergebenen Argumente enthält

int **func_get_args** (void)

In einem Array wird jedes Element zurück gegeben, das einer benutzer-definierten Funktion übergeben wurde. Sollten sie dies ausserhalb einer eigenen (benutzer-definierten) Funktion anwenden, wird eine Warnung ausgegeben.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Anzahl der Argumente: $numargs<br>\n";
    if ($numargs >= 2) {
        echo "Das 2. Argument ist: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
        echo "Argument $i ist: " . $arg_list[$i] . "<br>\n";
    }
}

foo (1, 2, 3);
?>
```

func_get_args() kann in Verbindung mit func_num_args() und func_get_arg() verwendet werden, um einer benutzer-definierten Funktion eine Argument-Liste mit variabler Länge zu übergeben.

Anmerkung: Diese Funktion steht ab PHP 4 zur Verfügung.

func_num_args (PHP 4 >= 4.0.0)

Gibt die Anzahl der einer Funktion übergebenen Argumente zurück

int **func_num_args** (void)

Die einer benutzer-definierten Funktion übergebene Anzahl an Argumenten wird zurück gegeben. Falls **func_num_args()** ausserhalb einer benutzer-definierten Funktion Verwendung findet, wird eine Warnung ausgegeben.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Anzahl der Argumente: $numargs\n";
}

foo (1, 2, 3);  // Ausgabe: 'Anzahl der Argumente: 3'
?>
```

func_num_args() kann in Verbindung mit func_get_arg() und func_get_args() benutzt werden, um einer benutzer-definerten Funktion eine variable Anzahl von Argumenten zu übergeben.

Anmerkung: Diese Funktion wurde in PHP 4 hinzu gefügt.

function_exists (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Falls die angegebene Funktion definiert ist, wird `TRUE` zurück gegeben

int **function_exists** (string function_name)

Prüft die Liste der definierten Funktionen auf das Vorhanden sein der Funktion mit dem Namen *`function_name`* . Sofern vorhanden wird `TRUE`, sonst `FALSE` zurück gegeben.

register_shutdown_function (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Registriert eine Funktion zur Ausführung beim Skript-Abschluss

int **register_shutdown_function** (string func)

Die mit *`func`* benannte Funktion wird im Skript zwecks Ausführung an dessen Ende eingetragen.

Bekannte Problemfälle:

Da innerhalb dieser Funktion keinerlei Ausgabe im Browser erzeugt werden, können sie die registrierte Funktion nicht mit print oder echo usw. debuggen.

XXXV. GNU Gettext

Die gettext Funktionen implementieren eine NLS (Native Language Support) API, welche zur Internationalisierung von PHP Anwendungen genutzt werden kann. Eine ausführliche Erklärung der folgenden Funktionen ist in der GNU gettext Dokumentation zu finden.

bindtextdomain (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Setzt den Pfad für einen Textbereich

string **bindtextdomain** (string domain, string directory)

Die Funktion **bindtextdomain()** setzt den Pfad für einen Textbereich.

dcgettext (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Setzt den Textbereich für eine Abfrage neu

string **dcgettext** (string domain, string message, int category)

Die Funktion **dcgettext()** erlaubt es, den Textbereich für die Wirkungsdauer von einem gettext() neu zu setzen. Man kann ausserdem eine Kategorie angeben.

dgettext (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Setzt den Textbereich für eine Abfrage neu

string **dgettext** (string domain, string message)

Die Funktion **dgettext()** erlaubt es, den Textbereich für die Wirkungsdauer von einem gettext() neu zu setzen.

gettext (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Liefert eine Übersetzung für den aktuellen Textbereich

string **gettext** (string message)

Diese Funktion gibt einen übersetzten String zurück, falls einer in der Übersetzungstabelle gefunden wird, andernfalls den übergebenen Originalstring. Als Alias für diese Funktion kann ein Unterstrich benutzt werden.

Beispiel 1. gettext()-check

```
<?php
// Sprache auf "German" setzen
putenv ("LANG=de");

// Pfad zu den Übersetzungstabellen setzen
bindtextdomain ("myPHPApp", "./locale");

// Textbereich auswählen
textdomain ("myPHPApp");

// Text ausgeben
```

```
print (gettext ("Welcome to My PHP Application"));
?>
```

textdomain (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Setzt den Standardbereich

int **textdomain** ([string library])

Diese Funktion setzt den für gettext() Aufrufe zu verwendenden Textbereich. Der vorherige Standardbereich wird zurück gegeben. Ein Aufruf ohne Parameter liefert den aktuellen Standardbereich ohne diesen zu ändern.

XXXVI. GMP functions

These functions allow you to work with arbitrary-length integers using the GNU MP library. In order to have these functions available, you must compile PHP with GMP support by using the `--with-gmp` option.

You can download the GMP library from http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

These functions have been added in PHP 4.0.4.

> **Anmerkung:** Most GMP functions accept GMP number arguments, defined as `resource` below. However, most of these functions will also accept numeric and string arguments, given that it is possible to convert the latter to a number. Also, if there is a faster function that can operate on integer arguments, it would be used instead of the slower function when the supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also the gmp_init() function.

> **Warnung**
>
> If you want to explicitly specify a large integer, specify it as a string. If you don't do that, PHP will interpret the integer-literal first, possibly resulting in loss of precision, even before `GMP` comes into play.

Beispiel 1. Factorial function using GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}

print gmp_strval (fact (1000)) . "\n";
?>
```

This will calculate factorial of 1000 (pretty big number) very fast.

gmp_init (PHP 4 >= 4.0.4)

Create GMP number

resource **gmp_init** (mixed number)

Creates a GMP number from an integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with `0x`.

Beispiel 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

Anmerkung: It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like gmp_add(). Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Convert GMP number to integer

int **gmp_intval** (resource gmpnumber)

This function allows to convert GMP number to integer.

Warnung

This function returns a useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use gmp_strval().

gmp_strval (PHP 4 >= 4.0.4)

Convert GMP number to string

string **gmp_strval** (resource gmpnumber [, int base])

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Beispiel 1. Converting a GMP number to a string

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_add (PHP 4 >= 4.0.4)

Add numbers

resource **gmp_add** (resource a, resource b)

Add two GMP numbers. The result will be a GMP number representing the sum of the arguments.

gmp_sub (PHP 4 >= 4.0.4)

Subtract numbers

resource **gmp_sub** (resource a, resource b)

Subtracts *b* from *a* and returns the result.

gmp_mul (PHP 4 >= 4.0.4)

Multiply numbers

resource **gmp_mul** (resource a, resource b)

Multiplies *a* by *b* and returns the result.

gmp_div_q (PHP 4 >= 4.0.4)

Divide numbers

resource **gmp_div_q** (resource a, resource b [, int round])

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards `+infinity`.
- *GMP_ROUND_MINUSINF*: The result is rounded towards `-infinity`.

This function can also be called as gmp_div().

See also gmp_div_r(), gmp_div_qr()

gmp_div_r (PHP 4 >= 4.0.4)

Remainder of the division of numbers

resource **gmp_div_r** (resource n, resource d [, int round])

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the gmp_div_q() function for description of the *round* argument.

See also gmp_div_q(), gmp_div_qr()

gmp_div_qr (PHP 4 >= 4.0.4)

Divide numbers and get quotient and remainder

array **gmp_div_qr** (resource n, resource d [, int round])

The function divides *n* by *d* and returns array, with the first element being `[n/d]` (the integer result of the division) and the second being `(n - [n/d] * d)` (the remainder of the division).

See the gmp_div_q() function for description of the *round* argument.

Beispiel 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also gmp_div_q(), gmp_div_r().

gmp_div (PHP 4 >= 4.0.4)

Divide numbers

resource **gmp_div** (resource a, resource b)

This function is an alias to gmp_div_q().

gmp_mod (PHP 4 >= 4.0.4)

Modulo operation

resource **gmp_mod** (resource n, resource d)

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

gmp_divexact (PHP 4 >= 4.0.4)

Exact division of numbers

resource **gmp_divexact** (resource n, resource d)

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

gmp_cmp (PHP 4 >= 4.0.4)

Compare numbers

int **gmp_cmp** (resource a, resource b)

Returns a positive value if `a > b`, zero if `a = b` and negative value if `a < b`.

gmp_neg (PHP 4 >= 4.0.4)

Negate number

resource **gmp_neg** (resource a)

Returns -*a*.

gmp_com (PHP 4 >= 4.0.4)

Calculates one's complement of a

resource **gmp_com** (resource a)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

gmp_abs (PHP 4 >= 4.0.4)

Absolute value

resource **gmp_abs** (resource a)

Returns absolute value of *a*.

gmp_sign (PHP 4 >= 4.0.4)

Sign of number

int **gmp_sign** (resource a)

Return sign of *a* - 1 if *a* is positive and -1 if it's negative.

gmp_fact (PHP 4 >= 4.0.4)

Factorial

resource **gmp_fact** (int a)

Calculates factorial (a!) of *a*.

gmp_sqrt (PHP 4 >= 4.0.4)

Square root

resource **gmp_sqrt** (resource a)

Calculates square root of *a*.

gmp_sqrtrm (unknown)

Square root with remainder

array **gmp_sqrtrm** (resource a)

Returns array where first element is the integer square root of *a* (see also gmp_sqrt()), and the second is the remainder (i.e., the difference between *a* and the first element squared).

gmp_perfect_square (PHP 4 >= 4.0.4)

Perfect square check

bool **gmp_perfect_square** (resource a)

Returns TRUE if *a* is a perfect square, FALSE otherwise.

See also: gmp_sqrt(), **gmp_sqrtrm()**.

gmp_pow (PHP 4 >= 4.0.4)

Raise number into power

resource **gmp_pow** (resource base, int exp)

Raise *base* into power *exp*. The case of 0^0 yields 1. *exp* cannot be negative.

gmp_powm (PHP 4 >= 4.0.4)

Raise number into power with modulo

resource **gmp_powm** (resource base, resource exp, resource mod)

Calculate (*base* raised into power *exp*) modulo *mod*. If *exp* is negative, result is undefined.

gmp_prob_prime (PHP 4 >= 4.0.4)

Check if number is "probably prime"

int **gmp_prob_prime** (resource a [, int reps])

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime. Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime.

The function uses Miller-Rabin’s probabilistic test.

gmp_gcd (PHP 4 >= 4.0.4)

Calculate GCD

resource **gmp_gcd** (resource a, resource b)

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of, or both, input operands are negative.

gmp_gcdext (PHP 4 >= 4.0.4)

Calculate GCD and multipliers

array **gmp_gcdext** (resource a, resource b)

Calculates g, s, and t, such that `a*s + b*t = g = gcd(a,b)`, where gcd is the greatest common divisor. Returns an array with respective elements g, s and t.

gmp_invert (PHP 4 >= 4.0.4)

Inverse by modulo

resource **gmp_invert** (resource a, resource b)

Computes the inverse of *a* modulo *b*. Returns `FALSE` if an inverse does not exist.

gmp_legendre (PHP 4 >= 4.0.4)

Legendre symbol

int **gmp_legendre** (resource a, resource p)

Compute the Legendre symbol (http://www.utm.edu/research/primes/glossary/LegendreSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_jacobi (PHP 4 >= 4.0.4)

Jacobi symbol

int **gmp_jacobi** (resource a, resource p)

Computes Jacobi symbol (http://www.utm.edu/research/primes/glossary/JacobiSymbol.html) of *a* and *p*. *p* should be odd and must be positive.

gmp_random (PHP 4 >= 4.0.4)

Random number

resource **gmp_random** (int limiter)

Generate a random number. The number will be between `limiter` and zero in value. If `limiter` is negative, negative numbers are generated.

gmp_and (PHP 4 >= 4.0.4)

Logical AND

resource **gmp_and** (resource a, resource b)

Calculates logical AND of two GMP numbers.

gmp_or (PHP 4 >= 4.0.4)

Logical OR

resource **gmp_or** (resource a, resource b)

Calculates logical inclusive OR of two GMP numbers.

gmp_xor (PHP 4 >= 4.0.4)

Logical XOR

resource **gmp_xor** (resource a, resource b)

Calculates logical exclusive OR (XOR) of two GMP numbers.

gmp_setbit (PHP 4 >= 4.0.4)

Set bit

resource **gmp_setbit** (resource &a, int index [, bool set_clear])

Sets bit `index` in `a`. `set_clear` defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_clrbit (PHP 4 >= 4.0.4)

Clear bit

resource **gmp_clrbit** (resource &a, int index)

Clears (sets to 0) bit *index* in *a*.

gmp_scan0 (PHP 4 >= 4.0.4)

Scan for 0

int **gmp_scan0** (resource a, int start)

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1 (PHP 4 >= 4.0.4)

Scan for 1

int **gmp_scan1** (resource a, int start)

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_popcount (PHP 4 >= 4.0.4)

Population count

int **gmp_popcount** (resource a)

Return the population count of *a*.

gmp_hamdist (PHP 4 >= 4.0.4)

Hamming distance

int **gmp_hamdist** (resource a, resource b)

Returns the hamming distance between *a* and *b*. Both operands should be non-negative.

XXXVII. HTTP-Funktionen

Diese Funktionen ermöglichen die Manipulation der an den Browser geschickten Informationen bis hinunter auf HTTP-Protokoll-Ebene.

header (PHP 3, PHP 4 >= 4.0.0)

Sendet einen HTTP-Header

int **header** (string string)

Die **header()**-Funktion wird zu Beginn einer HTML-Datei benutzt um HTTP-Anfangsinformationen zu senden. Siehe HTTP 1.1 Spezifikationen (http://www.w3.org/Protocols/rfc2616/rfc2616) für weitere Informationen. *Hinweis:* Beachten sie, dass die **header()**-Funktion aufgerufen werden muss, bevor irgendeine Ausgabe durch normale HTML-Tags oder PHP erzeugt wird. Es ist ein häufig anzutreffender Fehler, per include() Programm-Code einzubinden (bzw. per auto_prepend) oder Leerzeichen oder -zeilen vor dem Code stehen zu haben, die bereits Ausgaben bewirken (auch wenn diese leer sind) bevor **header()** aufgerufen wird.

Es gibt zwei Spezialfälle von Header-Aufrufen. Der Erste ist der "Location"-Header. Es wird nicht nur der Header an den Browser geschickt, der Apache erhält auch noch den Status-Code REDIRECT. Aus Sicht des Skript-Schreibers ist das nicht wichtig, aber für alle, die die Internas von Apache verstehen wollen, ist dies von Belang.

```
header ("Location: http://www.php.net");  /* Umleitung des Browsers
                                             zur PHP-Web-Seite. */
exit;  /* Sicher stellen, das nicht trotz Umleitung nachfolgender
          Code ausgeführt wird. */
```

Der zweite Spezialfall ist ein Header der mit der Zeichenfolge "HTTP/" beginnt (Groß-/Kleinschreibung ist egal). Falls sie z.B. die Dokument-Fehler-Anweisung 404 des Apache auf ein PHP-Skript zeigen lassen, ist es gut, wenn dieses Skript einen 404-Fehler erzeugt. In diesem Skript sollten sie also zuerst stehen haben:

```
header ("HTTP/1.0 404 Not Found");
```

PHP-Skripte erzeugen oft dynamisches HTML, das weder vom Browser noch von irgendeinem Proxy zwischen Web-Server und Client-Browser gepuffert ("gecached") werden soll bzw. darf. Bei vielen Proxies und Browsern kann das Cachen unterbunden werden und zwar mit:

```
header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT");    // Datum der
Vergangenheit
header ("Last-Modified: " . gmdate ("D, d M Y H:i:s") . " GMT");
                                                      // immer geändert
header ("Cache-Control: no-cache, must-revalidate");  // HTTP/1.1
header ("Pragma: no-cache");                          // HTTP/1.0
```

Siehe auch: headers_sent().

headers_sent (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gibt TRUE zurück, wenn die Header schon gesendet sind

boolean **headers_sent** (void)

Diese Funktion gibt TRUE zurück, wenn die HTTP-Header bereits abgeschickt worden sind, ansonsten wird FALSE zurück gegeben.

Siehe auch: header().

setcookie (PHP 3, PHP 4 >= 4.0.0)

Sendet ein Cookie

int **setcookie** (string name, string value, int expire, string path, string domain, int secure [, string value [, int expire [, string path [, string domain [, int secure]]]]])

setcookie() definiert ein mit den Header-Informationen zu übertragendes Cookie. Cookies müssen *vor* irgendwelchen anderen Headern gesendet werden (dies ist eine Einschränkung der Cookies, nicht von PHP). Sie müssen deshalb solche Funktionsaufrufe vor irgendwelchen <html>- oder <head>-Tags im Skript einsetzen.

Alle Argumente - ausser *name* - sind optional. Ist nur der Name angegeben wird das Cookie mit diesem Namen auf dem Client gelöscht. Sie können beliebige Argumente auch durch einen Leerstring (*""*) ersetzen, um diese Argumente zu übergehen. Der *expire*-Parameter und das *secure*-Argument sind Integer-Werte und können nicht mit einem Leerstring übersprungen werden. Setzen sie statt dessen NULL (*0*). Der *expire*-Parameter ist ein normaler UNIX-Zeitwert als Integer-Zahl, wie er von der time() oder mktime()-Funktion zurück gegeben wird. Das *secure*-Argument bedeutet, dass das Cookie nur über eine sichere HTTP-Verbindung geschickt werden soll.

Anwendungsbereiche:

- Cookies werden nicht sichtbar, bevor nicht eine Seite geladen wird, in der das Cookie sichtbar sein soll.
- Cookies müssen mit den selben Parametern gelöscht werden, mit denen sie gesetzt wurden.

In PHP3 werden mehrfache Aufrufe von **setcookie()** im selben Skript in umgekehrter Reihenfolge abgearbeitet. Sollten sie also ein Cookie löschen wollen bevor sie ein anderes setzen, sollten sie das Setzen vor dem Löschen vornehmen. In PHP4 werden mehrfache Aufrufe von **setcookie()** in der Reihenfolge ihres Aufrufs behandelt.

Einige Beispiele, wie Cookies gesetzt / gesendet werden:

Beispiel 1. setcookie()-Beispiele:

```
setcookie ("TestCookie", "Test Value");
setcookie ("TestCookie", $value, time()+3600);  /* verfällt in 1 Stunde */
setcookie ("TestCookie", $value, time()+3600, "/~rasmus/", ".utoronto.ca",1);
```

Beispiele, wie die vorhergehenden Cookies wieder gelöscht werden:

Beispiel 2. setcookie()-Beispiele:

```
setcookie ("TestCookie");
// Setzen des Ablauf-Zeitpunktes auf 1 Stunde vorher
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".utoronto.ca",1);
```

Beim Löschen eines Cookies sollten sie sicher stellen, dass die Verfallszeit in der Vergangenheit liegt, damit ihr Browser "zufrieden" ist.

Beachten sie, dass der Wertebereich des Cookies automatisch URL-konform codiert (urlencoded) wird, sobald sie das Cookie senden, und wenn es gelesen wird, wird es automatisch URL-konform decodiert und einer Variablen zugewiesen, die den selben Namen wie das Cookie trägt. Um die Inhalte unserer Test-Cookies zu sehen, brauchen sie nur eines der folgenden Beispiel zu benutzen:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS["TestCookie"];
```

Sie können auch ein Array an Cookies setzen, in dem sie die Array-Schreibweise benutzen. Dadurch werden so viele Cookies gesetzt, wie ihr Array Elemente hat. Sobald das Cookie aber von ihrem Skript gelesen wird, werden alle Werte in ein einziges Array mit dem Cookie-Namen eingelesen:

```
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");
if (isset ($cookie)) {
    while (list ($name, $value) = each ($cookie)) {
        echo "$name == $value<br>\n";
    }
}
```

Weitere Informationen zu Cookies und deren Spezifikationen erhalten sie bei Netscape unter http://www.netscape.com/newsref/std/cookie_spec.html.

Microsofts Internet-Explorer 4 mit Servive-Pack 1 geht nicht korrekt mit Cookies um, die den Pfad-Parameter beinhalten.

Netscape Communicator 4.05 und Microsoft Internet Explorer 3.x scheinen mit Cookies Probleme zu haben, wenn die Argumente für Pfad und Zeit nicht angegeben sind.

XXXVIII. Hyperwave functions

Introduction

Hyperwave has been developed at IICM (http://www.iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 4.1, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user. An attribute is a name/value pair of the form name=value. The complete object record contains as many of those pairs as the user likes. The name of an attribute does not have to be unique, e.g. a title may appear several times within an object record. This makes sense if you want to specify a title in several languages. In such a case there is a convention, that each title value is preceded by the two letter language abbreviation followed by a colon, e.g. 'en:Title in English' or 'ge:Titel in deutsch'. Other attributes like a description or keywords are potential candidates. You may also replace the language abbreviation by any other string as long as it separated by colon from the rest of the attribute value.

Each object record has native a string representation with each name/value pair separated by a newline. The Hyperwave extension also knows a second representation which is an associated array with the attribute name being the key. Multilingual attribute values itself form another associated array with the key being the language abbreviation. Actually any multiple attribute forms an associated array with the string left to the colon in the attribute value being the key. (This is not fully implemented. Only the attributes Title, Description and Keyword are treated properly yet.)

Besides the documents, all hyper links contained in a document are stored as object records as well. Hyper links which are in a document will be removed from it and stored as individual objects, when the document is inserted into the database. The object record of the link contains information about where it starts and where it ends. In order to gain the original document you will have to retrieve the plain document without the links and the list of links and reinsert them (The functions hw_pipedocument() and hw_gettext() do this for you. The advantage of separating links from the document is obvious. Once a document to which a link is pointing to changes its name, the link can easily be modified accordingly. The document containing the link is not affected at all. You may even add a link to a document without modifying the document itself.

Saying that hw_pipedocument() and hw_gettext() do the link insertion automatically is not as simple as it sounds. Inserting links implies a certain hierarchy of the documents. On a web server this is given by the file system, but Hyperwave has its own hierarchy and names do not reflect the position of an object in that hierarchy. Therefore creation of links first of all requires a mapping from the Hyperwave hierarchy and namespace into a web hierarchy respective web namespace. The fundamental difference between Hyperwave and the web is the clear distinction between names and hierarchy in Hyperwave. The name does not contain any information about the objects position in the hierarchy. In the web the name also contains the information on where the object is located in the hierarchy. This leads to two possibles ways of mapping. Either the Hyperwave hierarchy and name of the Hyperwave object is reflected in the URL or the name only. To make things simple the second approach is used. Hyperwave object with name 'my_object' is mapped to 'http://host/my_object'

disregarding where it resides in the Hyperwave hierarchy. An object with name 'parent/my_object' could be the child of 'my_object' in the Hyperwave hierarchy, though in a web namespace it appears to be just the opposite and the user might get confused. This can only be prevented by selecting reasonable object names.

Having made this decision a second problem arises. How do you involve PHP? The URL http://host/my_object will not call any PHP script unless you tell your web server to rewrite it to e.g. 'http://host/php3_script/my_object' and the script 'php3_script' evaluates the $PATH_INFO variable and retrieves the object with name 'my_object' from the Hyperwave server. Their is just one little drawback which can be fixed easily. Rewriting any URL would not allow any access to other document on the web server. A PHP script for searching in the Hyperwave server would be impossible. Therefore you will need at least a second rewriting rule to exclude certain URLS like all e.g. starting with http://host/Hyperwave. This is basically sharing of a namespace by the web and Hyperwave server.

Based on the above mechanism links are insert into documents.

It gets more complicated if PHP is not run as a server module or CGI script but as a standalone application e.g. to dump the content of the Hyperwave server on a CD-ROM. In such a case it makes sense to retain the Hyperwave hierarchy and map in onto the file system. This conflicts with the object names if they reflect its own hierarchy (e.g. by choosing names including '/'). Therefore '/' has to be replaced by another character, e.g. '_'. to be continued.

The network protocol to communicate with the Hyperwave server is called HG-CSP (http://www.hyperwave.com/7.17-hg-prot) (Hyper-G Client/Server Protocol). It is based on messages to initiate certain actions, e.g. get object record. In early versions of the Hyperwave Server two native clients (Harmony, Amadeus) were provided for communication with the server. Those two disappeared when Hyperwave was commercialised. As a replacement a so called wavemaster was provided. The wavemaster is like a protocol converter from HTTP to HG-CSP. The idea is to do all the administration of the database and visualisation of documents by a web interface. The wavemaster implements a set of placeholders for certain actions to customise the interface. This set of placeholders is called the PLACE Language. PLACE lacks a lot of features of a real programming language and any extension to it only enlarges the list of placeholders. This has led to the use of JavaScript which IMO does not make life easier.

Adding Hyperwave support to PHP should fill in the gap of a missing programming language for interface customisation. It implements all the messages as defined by the HG-CSP but also provides more powerful commands to e.g. retrieve complete documents.

Hyperwave has its own terminology to name certain pieces of information. This has widely been taken over and extended. Almost all functions operate on one of the following data types.

- object ID: An unique integer value for each object in the Hyperwave server. It is also one of the attributes of the object record (ObjectID). Object ids are often used as an input parameter to specify an object.
- object record: A string with attribute-value pairs of the form attribute=value. The pairs are separated by a carriage return from each other. An object record can easily be converted into an object array with **hw_object2array()**. Several functions return object records. The names of those functions end with obj.
- object array: An associated array with all attributes of an object. The key is the attribute name. If an attribute occurs more than once in an object record it will result in another indexed or associated array. Attributes which are language depended (like the title, keyword, description) will form an associated array with the key set to the language abbreviation. All other multiple

attributes will form an indexed array. PHP functions never return object arrays.

- hw_document: This is a complete new data type which holds the actual document, e.g. HTML, PDF etc. It is somewhat optimised for HTML documents but may be used for any format.

Several functions which return an array of object records do also return an associated array with statistical information about them. The array is the last element of the object record array. The statistical array contains the following entries:

Hidden

Number of object records with attribute PresentationHints set to Hidden.

CollectionHead

Number of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHead

Number of object records with attribute PresentationHints set to FullCollectionHead.

CollectionHeadNr

Index in array of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHeadNr

Index in array of object records with attribute PresentationHints set to FullCollectionHead.

Total

Total: Number of object records.

Integration with Apache

The Hyperwave extension is best used when PHP is compiled as an Apache module. In such a case the underlying Hyperwave server can be hidden from users almost completely if Apache uses its rewriting engine. The following instructions will explain this.

Since PHP with Hyperwave support built into Apache is intended to replace the native Hyperwave solution based on Wavemaster I will assume that the Apache server will only serve as a Hyperwave web interface. This is not necessary but it simplifies the configuration. The concept is quite simple. First of all you need a PHP script which evaluates the PATH_INFO variable and treats its value as the name of a Hyperwave object. Let's call this script 'Hyperwave'. The URL http://your.hostname/Hyperwave/name_of_object would than return the Hyperwave object with the name 'name_of_object'. Depending on the type of the object the script has to react accordingly. If it is a collection, it will probably return a list of children. If it is a document it will return the mime type and the content. A slight improvement can be achieved if the Apache rewriting engine is used. From the users point of view it would be more straight forward if the URL http://your.hostname/name_of_object would return the object. The rewriting rule is quite easy:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Now every URL relates to an object in the Hyperwave server. This causes a simple to solve problem. There is no way to execute a different script, e.g. for searching, than the 'Hyperwave' script. This can be fixed with another rewriting rule like the following:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

This will reserve the directory `/usr/local/apache/htdocs/hw` for additional scripts and other files. Just make sure this rule is evaluated before the one above. There is just a little drawback: all Hyperwave objects whose name starts with 'hw/' will be shadowed. So, make sure you don't use such names. If you need more directories, e.g. for images just add more rules or place them all in one directory. Finally, don't forget to turn on the rewriting engine with

```
RewriteEngine on
```

My experiences have shown that you will need the following scripts:

- to return the object itself
- to allow searching
- to identify yourself
- to set your profile
- one for each additional function like to show the object attributes, to show information about users, to show the status of the server, etc.

Todo

There are still some things todo:

- The hw_InsertDocument has to be split into hw_insertobject() and **hw_putdocument()**.
- The names of several functions are not fixed, yet.
- Most functions require the current connection as its first parameter. This leads to a lot of typing, which is quite often not necessary if there is just one open connection. A default connection will improve this.
- Conversion form object record into object array needs to handle any multiple attribute.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

convert attributes from object array to object record

strin **hw_array2objrec** (array object_array)

Converts an *object_array* into an object record. Multiple attributes like 'Title' in different languages are treated properly.

See also hw_objrec2array().

hw_Children (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of children

array **hw_children** (int connection, int objectID)

Returns an array of object ids. Each id belongs to a child of the collection with ID *objectID*. The array contains all children both documents and collections.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of children

array **hw_childrenobj** (int connection, int objectID)

Returns an array of object records. Each object record belongs to a child of the collection with ID *objectID*. The array contains all children both documents and collections.

hw_Close (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

closes the Hyperwave connection

int **hw_close** (int connection)

Returns FALSE if connection is not a valid connection index, otherwise TRUE. Closes down the connection to a Hyperwave server with the given connection index.

hw_Connect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

opens a connection

int **hw_connect** (string host, int port, string username, string password)

Opens a connection to a Hyperwave server and returns a connection index on success, or FALSE if the connection could not be made. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a

case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple connections open at once. Keep in mind, that the password is not encrypted.

See also hw_pconnect().

hw_Cp (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

copies objects

int **hw_cp** (int connection, array object_id_array, int destination id)

Copies the objects with object ids as specified in the second parameter to the collection with the id *destination id*.

The value return is the number of copied objects.

See also hw_mv().

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

deletes object

int **hw_deleteobject** (int connection, int object_to_delete)

Deletes the object with the given object id in the second parameter. It will delete all instances of the object.

Returns TRUE if no error occurs otherwise FALSE.

See also hw_mv().

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object id object belonging to anchor

int **hw_docbyanchor** (int connection, int anchorID)

Returns an th object id of the document to which *anchorID* belongs.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object record object belonging to anchor

string **hw_docbyanchorobj** (int connection, int anchorID)

Returns an th object record of the document to which *anchorID* belongs.

hw_Document_Attributes (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object record of hw_document

string **hw_document_attributes** (int hw_document)

Returns the object record of the document.

For backward compatibility, **hw_documentattributes()** is also accepted. This is deprecated, however.

See also hw_document_bodytag(), and hw_document_size().

hw_Document_BodyTag (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

body tag of hw_document

string **hw_document_bodytag** (int hw_document)

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also hw_document_attributes(), and hw_document_size().

For backward compatibility, **hw_documentbodytag()** is also accepted. This is deprecated, however.

hw_Document_Content (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

returns content of hw_document

string **hw_document_content** (int hw_document)

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also hw_document_attributes(), hw_document_size(), and hw_document_setcontent().

hw_Document_SetContent (PHP 4 >= 4.0.0)

sets/replaces content of hw_document

string **hw_document_setcontent** (int hw_document, string content)

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also hw_document_attributes(), hw_document_size(), and hw_document_content().

hw_Document_Size (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

size of hw_document

int **hw_document_size** (int hw_document)

Returns the size in bytes of the document.

See also hw_document_bodytag(), and hw_document_attributes().

For backward compatibility, **hw_documentsize()** is also accepted. This is deprecated, however.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

returns error message

string **hw_errormsg** (int connection)

Returns a string containing the last error message or 'No Error'. If FALSE is returned, this function failed. The message relates to the last command.

hw_EditText (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

retrieve text document

int **hw_edittext** (int connection, int hw_document)

Uploads the text document to the server. The object record of the document may not be modified while the document is edited. This function will only works for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also hw_pipedocument(), hw_free_document(), hw_document_bodytag(), hw_document_size(), hw_output_document(), hw_gettext().

hw_Error (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

error number

int **hw_error** (int connection)

Returns the last error number. If the return value is 0 no error has occurred. The error relates to the last command.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

frees hw_document

int **hw_free_document** (int hw_document)

Frees the memory occupied by the Hyperwave document.

hw_GetParents (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of parents

array **hw_getparents** (int connection, int objectID)

Returns an indexed array of object ids. Each object id belongs to a parent of the object with ID `objectID`.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of parents

array **hw_getparentsobj** (int connection, int objectID)

Returns an indexed array of object records plus an associated array with statistical information about the object records. The associated array is the last entry of the returned array. Each object record belongs to a parent of the object with ID `objectID`.

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of child collections

array **hw_getchildcoll** (int connection, int objectID)

Returns an array of object ids. Each object ID belongs to a child collection of the collection with ID `objectID`. The function will not return child documents.

See also hw_children(), and hw_getchilddoccoll().

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of child collections

array **hw_getchildcollobj** (int connection, int objectID)

Returns an array of object records. Each object records belongs to a child collection of the collection with ID `objectID`. The function will not return child documents.

See also hw_childrenobj(), and hw_getchilddoccollobj().

hw_GetRemote (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Gets a remote document

int **hw_getremote** (int connection, int objectID)

Returns a remote document. Remote documents in Hyperwave notation are documents retrieved from an external source. Common remote documents are for example external web pages or queries in a database. In order to be able to access external sources throught remote documents Hyperwave introduces the HGI (Hyperwave Gateway Interface) which is similar to the CGI. Currently, only ftp, http-servers and some databases can be accessed by the HGI. Calling **hw_getremote()** returns the document from the external source. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also hw_getremotechildren().

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Gets children of remote document

int **hw_getremotechildren** (int connection, string object record)

Returns the children of a remote document. Children of a remote document are remote documents itself. This makes sense if a database query has to be narrowed and is explained in Hyperwave Programmers' Guide. If the number of children is 1 the function will return the document itself formated by the Hyperwave Gateway Interface (HGI). If the number of children is greater than 1 it will return an array of object record with each maybe the input value for another call to **hw_getremotechildren()**. Those object records are virtual and do not exist in the Hyperwave server, therefore they do not have a valid object ID. How exactely such an object record looks like is up to the HGI. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also hw_getremote().

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns anchors pointing at object

array **hw_getsrcbydestobj** (int connection, int objectID)

Returns the object records of all anchors pointing to the object with ID `objectID`. The object can either be a document or an anchor of type destination.

See also hw_getanchors().

hw_GetObject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object record

array **hw_getobject** (int connection, [int|array] objectID, string query)

Returns the object record for the object with ID `objectID` if the second parameter is an integer. If the second parameter is an array of integer the function will return an array of object records. In such a case the last parameter is also evaluated which is a query string.

The query string has the following syntax:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NOT */

<expr> "||" <expr> | /* OR */

<expr> "&&" <expr> | /* AND */

<attribute> <operator> <value>

<attribute> ::= /* any attribute name (Title, Author, DocumentType ...) */

<operator> ::= "=" | /* equal */

"<" | /* less than (string compare) */

">" | /* greater than (string compare) */

"~" /* regular expression matching */

The query allows to further select certain objects from the list of given objects. Unlike the other query functions, this query may use not indexed attributes. How many object records are returned depends on the query and if access to the object is allowed.

See also hw_getandlock(), and hw_getobjectbyquery().

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

return bject record and lock object

string **hw_getandlock** (int connection, int objectID)

Returns the object record for the object with ID `objectID`. It will also lock the object, so other users cannot access it until it is unlocked.

See also hw_unlock(), and hw_getobject().

hw_GetText (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

retrieve text document

int **hw_gettext** (int connection, int objectID [, mixed rootID/prefix])

Returns the document with object ID `objectID`. If the document has anchors which can be inserted, they will be inserted already. The optional parameter `rootID/prefix` can be a string or

an integer. If it is an integer it determines how links are inserted into the document. The default is 0 and will result in links that are constructed from the name of the link's destination object. This is useful for web applications. If a link points to an object with name 'internet_movie' the HTML link will be <A HREF="/internet_movie">. The actual location of the source and destination object in the document hierachy is disregarded. You will have to set up your web browser, to rewrite that URL to for example '/my_script.php3/internet_movie'. 'my_script.php3' will have to evaluate $PATH_INFO and retrieve the document. All links will have the prefix '/my_script.php3/'. If you do not want this you can set the optional parameter *rootID/prefix* to any prefix which is used instead. Is this case it has to be a string.

If *rootID/prefix* is an integer and unequal to 0 the link is constructed from all the names starting at the object with the id *rootID/prefix* separated by a slash relative to the current object. If for example the above document 'internet_movie' is located at 'a-b-c-internet_movie' with '-' being the seperator between hierachy levels on the Hyperwave server and the source document is located at 'a-b-d-source' the resulting HTML link would be: <A HREF="../c/internet_movie">. This is useful if you want to download the whole server content onto disk and map the document hierachy onto the file system.

This function will only work for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also hw_pipedocument(), hw_free_document(), hw_document_bodytag(), hw_document_size(), and hw_output_document().

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object

array **hw_getobjectbyquery** (int connection, string query, int max_hits)

Searches for objects on the whole server and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyqueryobj().

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object

array **hw_getobjectbyqueryobj** (int connection, string query, int max_hits)

Searches for objects on the whole server and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquery().

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object in collection

array **hw_getobjectbyquerycoll** (int connection, int objectID, string query, int max_hits)

Searches for objects in collection with ID *objectID* and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquerycollobj().

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object in collection

array **hw_getobjectbyquerycollobj** (int connection, int objectID, string query, int max_hits)

Searches for objects in collection with ID *objectID* and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquerycoll().

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of child documents of collection

array **hw_getchilddoccoll** (int connection, int objectID)

Returns array of object ids for child documents of a collection.

See also hw_children(), and hw_getchildcoll().

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of child documents of collection

array **hw_getchilddoccollobj** (int connection, int objectID)

Returns an array of object records for child documents of a collection.

See also hw_childrenobj(), and hw_getchildcollobj().

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of anchors of document

array **hw_getanchors** (int connection, int objectID)

Returns an array of object ids with anchors of the document with object ID `objectID`.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of anchors of document

array **hw_getanchorsobj** (int connection, int objectID)

Returns an array of object records with anchors of the document with object ID `objectID`.

hw_Mv (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

moves objects

int **hw_mv** (int connection, array object id array, int source id, int destination id)

Moves the objects with object ids as specified in the second parameter from the collection with id `source id` to the collection with the id `destination id`. If the destination id is 0 the objects will be unlinked from the source collection. If this is the last instance of that object it will be deleted. If you want to delete all instances at once, use hw_deleteobject().

The value return is the number of moved objects.

See also hw_cp(), and hw_deleteobject().

hw_Identify (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

identifies as user

int **hw_identify** (string username, string password)

Identifies as user with `username` and `password`. Identification is only valid for the current session. I do not thing this function will be needed very often. In most cases it will be easier to identify with the opening of the connection.

See also hw_connect().

hw_InCollections (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

check if object ids in collections

array **hw_incollections** (int connection, array object_id_array, array collection_id_array, int return_collections)

Checks whether a set of objects (documents or collections) specified by the `object_id_array` is part of the collections listed in `collection_id_array`. When the fourth parameter `return_collections` is 0, the subset of object ids that is part of the collections (i.e., the documents or collections that are children of one or more collections of collection ids or their subcollections, recursively) is returned as an array. When the fourth parameter is 1, however, the set of collections that have one or more objects of this subset as children are returned as an array. This option allows a client to, e.g., highlight the part of the collection hierarchy that contains the matches of a previous query, in a graphical overview.

hw_Info (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

info about connection

string **hw_info** (int connection)

Returns information about the current connection. The returned string has the following format: <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

insert collection

int **hw_inscoll** (int connection, int objectID, array object_array)

Inserts a new collection with attributes as in `object_array` into collection with object ID `objectID`.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

insert document

int **hw_insdoc** (int connection, int parentID, string object_record, string text)

Inserts a new document with attributes as in `object_record` into collection with object ID `parentID`. This function inserts either an object record only or an object record and a pure ascii text in `text` if `text` is given. If you want to insert a general document of any kind use hw_insertdocument() instead.

See also hw_insertdocument(), and hw_inscoll().

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

upload any document

int **hw_insertdocument** (int connection, int parent_id, int hw_document)

Uploads a document into the collection with `parent_id`. The document has to be created before with hw_new_document(). Make sure that the object record of the new document contains at least the attributes: Type, DocumentType, Title and Name. Possibly you also want to set the MimeType. The functions returns the object id of the new document or `FALSE`.

See also hw_pipedocument().

hw_InsertObject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

inserts an object record

int **hw_insertobject** (int connection, string object rec, string parameter)

Inserts an object into the server. The object can be any valid hyperwave object. See the HG-CSP documentation for a detailed information on how the parameters have to be.

Note: If you want to insert an Anchor, the attribute Position has always been set either to a start/end value or to 'invisible'. Invisible positions are needed if the annotation has no correspondig link in the annotation text.

See also hw_pipedocument(), hw_insertdocument(), hw_insdoc(), and hw_inscoll().

hw_mapid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Maps global id on virtual local id

int **hw_mapid** (int connection, int server id, int object id)

Maps a global object id on any hyperwave server, even those you did not connect to with hw_connect(), onto a virtual object id. This virtual object id can then be used as any other object id, e.g. to obtain the object record with hw_getobject(). The server id is the first part of the global object id (GOid) of the object which is actually the IP number as an integer.

Note: In order to use this function you will have to set the F_DISTRIBUTED flag, which can currently only be set at compile time in hg_comm.c. It is not set by default. Read the comment at the beginning of hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

modifies object record

int **hw_modifyobject** (int connection, int object_to_change, array remove, array add, int mode)

This command allows to remove, add, or modify individual attributes of an object record. The object is specified by the Object ID `object_to_change`. The first array `remove` is a list of attributes to remove. The second array `add` is a list of attributes to add. In order to modify an attribute one will have to remove the old one and add a new one. **hw_modifyobject()** will always remove the attributes before it adds attributes unless the value of the attribute to remove is not a string or array.

The last parameter determines if the modification is performed recursively. 1 means recurive modification. If some of the objects cannot be modified they will be skiped without notice. hw_error() may not indicate an error though some of the objects could not be modified.

The keys of both arrays are the attributes name. The value of each array element can either be an array, a string or anything else. If it is an array each attribute value is constructed by the key of each element plus a colon and the value of each element. If it is a string it is taken as the attribute value. An empty string will result in a complete removal of that attribute. If the value is neither a string nor an array but something else, e.g. an integer, no operation at all will be performed on the attribute. This is neccessary if you want to to add a completely new attribute not just a new value for an existing attribute. If the remove array contained an empty string for that attribute, the attribute would be tried to be removed which would fail since it doesn't exist. The following addition of a new value for that attribute would also fail. Setting the value for that attribute to e.g. 0 would not even try to remove it and the addition will work.

If you would like to change the attribute 'Name' with the current value 'books' into 'articles' you will have to create two arrays and call **hw_modifyobject()**.

Beispiel 1. modifying an attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => "books");
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

In order to delete/add a name=value pair from/to the object record just pass the remove/add array and set the last/third parameter to an empty array. If the attribute is the first one with that name to add, set attribute value in the remove array to an integer.

Beispiel 2. adding a completely new attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => 0);
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Anmerkung: Multilingual attributes, e.g. 'Title', can be modified in two ways. Either by providing the attributes value in its native form 'language':'title' or by providing an array with elements for each language as described above. The above example would than be:

Beispiel 3. modifying Title attribute

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or

Beispiel 4. modifying Title attribute

```
$remarr = array("Title" => array("en" => "Books"));
$addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This removes the english title 'Books' and adds the english title 'Articles' and the german title 'Artikel'.

Beispiel 5. removing attribute

```
$remarr = array("Title" => "");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Anmerkung: This will remove all attributes with the name 'Title' and adds a new 'Title' attribute. This comes in handy if you want to remove attributes recursively.

Anmerkung: If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.

Anmerkung: Only the attributes 'Title', 'Description' and 'Keyword' will properly handle the language prefix. If those attributes don't carry a language prefix, the prefix 'xx' will be assigned.

Anmerkung: The 'Name' attribute is somewhat special. In some cases it cannot be complete removed. You will get an error message 'Change of base attribute' (not clear when this happens). Therefore you will always have to add a new Name first and than remove the old one.

Anmerkung: You may not suround this function by calls to hw_getandlock() and hw_unlock(). **hw_modifyobject()** does this internally.

Returns `TRUE` if no error occurs otherwise `FALSE`.

hw_New_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

create new document

int **hw_new_document** (string object_record, string document_data, int document_size)

Returns a new Hyperwave document with document data set to *`document_data`* and object record set to *`object_record`*. The length of the *`document_data`* has to passed in *`document_size`*This function does not insert the document into the Hyperwave server.

See also hw_free_document(), hw_document_size(), hw_document_bodytag(), hw_output_document(), and hw_insertdocument().

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

convert attributes from object record to object array

array **hw_objrec2array** (string object_record [, array format])

Converts an *`object_record`* into an object array. The keys of the resulting array are the attributes names. Multi-value attributes like 'Title' in different languages form its own array. The keys of this array are the left part to the colon of the attribute value. This left part must be two characters long. Other multi-value attributes without a prefix form an indexed array. If the optional parameter is missing the attributes 'Title', 'Description' and 'Keyword' are treated as language attributes and the attributes 'Group', 'Parent' and 'HtmlAttr' as non-prefixed multi-value attributes. By passing an array holding the type for each attribute you can alter this behaviour. The array is an associated array with the attribute name as its key and the value being one of `HW_ATTR_LANG` or `HW_ATTR_NONE`

See also hw_array2objrec().

hw_Output_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

prints hw_document

int **hw_output_document** (int hw_document)

Prints the document without the BODY tag.

For backward compatibility, **hw_outputdocument()** is also accepted. This is deprecated, however.

hw_pConnect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

make a persistent database connection

int **hw_pconnect** (string host, int port, string username, string password)

Returns a connection index on success, or `FALSE` if the connection could not be made. Opens a persistent connection to a Hyperwave server. Each of the arguments should be a quoted string, except for the port number. The *`username`* and *`password`* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple persistent connections open at once.

See also hw_connect().

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

retrieve any document

int **hw_pipedocument** (int connection, int objectID)

Returns the Hyperwave document with object ID *`objectID`*. If the document has anchors which can be inserted, they will have been inserted already. The document will be transfered via a special data connection which does not block the control connection.

See also hw_gettext() for more on link insertion, hw_free_document(), hw_document_size(), hw_document_bodytag(), and hw_output_document().

hw_Root (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

root object id

int **hw_root** ( )

Returns the object ID of the hyperroot collection. Currently this is always 0. The child collection of the hyperroot is the root collection of the connected server.

hw_Unlock (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

unlock object

int **hw_unlock** (int connection, int objectID)

Unlocks a document, so other users regain access.

See also hw_getandlock().

hw_Who (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

List of currently logged in users

int **hw_who** (int connection)

Returns an array of users currently logged into the Hyperwave server. Each entry in this array is an array itself containing the elements id, name, system, onSinceDate, onSinceTime, TotalTime and self. 'self' is 1 if this entry belongs to the user who initianted the request.

hw_getusername (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

name of currently logged in user

string **hw_getusername** (int connection)

Returns the username of the connection.

hw_stat (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns status string

string **hw_stat** (int link)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

hw_setlinkroot (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Set the id to which links are calculated

void **hw_setlinkroot** (int link, int rootid)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

hw_connection_info (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Prints information about the connection to Hyperwave server

void **hw_connection_info** (int link)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

hw_dummy (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Hyperwave dummy function

string **hw_dummy** (int link, int id, int msgid)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

hw_insertanchors (PHP 4 >= 4.0.4)

Inserts only anchors into text

string **hw_insertanchors** (int hwdoc, array anchorecs, array dest [, array urlprefixes])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

hw_getrellink (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Get link from source to dest relative to rootid

string **hw_getrellink** (int link, int rootid, int sourceid, int destid)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

hw_changeobject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Changes attributes of an object (obsolete)

void **hw_changeobject** (int link, int objid, array attributes)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XXXIX. ICAP Functions [deprecated]

To get these functions to work, you have to compile PHP with `--with-icap`. That requires the icap library to be installed, which is not longer supported and available.

> **Anmerkung:** Icap will be removed in near future. Neither this module, nor those versions of icap library are supported any longer. If you want to use calendar capabilities in php, use mcal instead.

icap_open (PHP 4 >= 4.0.0)

Opens up an ICAP connection

stream **icap_open** (string calendar, string username, string password, string options)

Returns an ICAP stream on success, `FALSE` on error.

icap_open() opens up an ICAP connection to the specified *`calendar`* store. If the optional *`options`* is specified, passes the *`options`* to that mailbox also.

icap_close (unknown)

Close an ICAP stream

int **icap_close** (int icap_stream [, int flags])

Closes the given icap stream.

icap_fetch_event (PHP 4 >= 4.0.0)

Fetches an event from the calendar stream/

int **icap_fetch_event** (int stream_id, int event_id [, int options])

icap_fetch_event() fetches an event from the calendar stream specified by *`event_id`*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - `TRUE` if the event if public, `FALSE` if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_list_events (PHP 4 >= 4.0.0)

Return a list of events between two given datetimes

array **icap_list_events** (int stream_id, int begin_date [, int end_date])

Returns an array of event ID's that are between the two given datetimes.

icap_list_events() function takes in a beginning datetime and an end datetime for a calendar stream. An array of event id's that are between the given datetimes are returned.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_store_event (PHP 4 >= 4.0.0)

Store an event into an ICAP calendar

string **icap_store_event** (int stream_id, object event)

icap_store_event() Stores an event into an ICAP calendar. An event object consists of:

- int public - 1 if public, 0 if private;
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - Number of minutes before the event to sned out an alarm.
- datetime start - datetime object of the start of the event.
- datetime end - datetime object of the end of the event.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month

- int hour - hour
- int min - minutes
- int sec - seconds

Returns TRUE on success and FALSE on error.

icap_delete_event (PHP 4 >= 4.0.0)

Delete an event from an ICAP calendar

string **icap_delete_event** (int stream_id, int uid)

icap_delete_event() deletes the calendar event specified by the *uid*.

Returns TRUE.

icap_snooze (PHP 4 >= 4.0.0)

Snooze an alarm

string **icap_snooze** (int stream_id, int uid)

icap_snooze() turns on an alarm for a calendar event specified by the *uid*.

Returns TRUE.

icap_list_alarms (PHP 4 >= 4.0.0)

Return a list of events that has an alarm triggered at the given datetime

int **icap_list_alarms** (int stream_id, array date, array time)

Returns an array of event ID's that has an alarm going off at the given datetime.

icap_list_alarms() function takes in a datetime for a calendar stream. An array of event id's that has an alarm should be going off at the datetime are returned.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_create_calendar (PHP 4 >= 4.0.0)

Create a new calendar

string **icap_create_calendar** (int stream_id, string calendar)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

icap_rename_calendar (PHP 4 >= 4.0.0)

Rename a calendar

string **icap_rename_calendar** (int stream_id, string old_name, string new_name)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

icap_delete_calendar (PHP 4 >= 4.0.0)

Delete a calendar

string **icap_delete_calendar** (int stream_id, string calendar)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

icap_reopen (PHP 4 >= 4.0.0)

Reopen ICAP stream to new calendar

int **icap_reopen** (int stream_id, string calendar [, int options])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XL. iconv functions

This module contains an interface to the iconv library functions. To be able to use the functions defined in this module you must compile you PHP interpreter using the --with-iconv option. In order to do so, you must have iconv() function in standard C library or libiconv installed on your system. libiconv library is available from http://www.gnu.org/software/libiconv/ (http://www.gnu.org/software/libiconv/).

iconv library function converts files between various encoded character sets. The supported character sets depend on iconv() implementation on your system. Note that iconv() function in some system is not work well as you expect. In this case, you should install libiconv library.

iconv (PHP 4 >= 4.0.5)

Convert string to requested character encoding

string **iconv** (string in_charset, string out_charset, string str)

It converts the string *string* encoded in *in_charset* to the string encoded in *out_charset*. It returns the converted string or FALSE, if it fails.

Beispiel 1. iconv() example:

```
echo iconv("ISO-8859-1","UTF-8","This is test.");
```

iconv_get_encoding (PHP 4 >= 4.0.5)

Get current setting for character encoding conversion

array **iconv_get_encoding** ([string type])

It returns the current settings of ob_iconv_handler() as array or FALSE in failure.

See also: iconv_set_encoding() and ob_iconv_handler().

iconv_set_encoding (PHP 4 >= 4.0.5)

Set current setting for character encoding conversion

array **iconv_set_encoding** (string type, string charset)

It changes the value of *type* to *charset* and returns TRUE in success or FALSE in failure.

Beispiel 1. iconv_set_encoding() example:

```
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
```

See also: iconv_get_encoding() and ob_iconv_handler().

ob_iconv_handler (PHP 4 >= 4.0.5)

Convert character encoding as output buffer handler

array **ob_iconv_handler** (string contents, int status)

It converts the string encoded in *internal_encoding* to *output_encoding*.

internal_encoding and *output_encoding* should be defined by iconv_set_encoding() or in configuration file.

Beispiel 1. ob_iconv_handler() example:

```
ob_start("ob_iconv_handler"); // start output buffering
```

See also: iconv_get_encoding() and iconv_set_encoding().

XLI. Grafik-Funktionen

Sie können die Grafik-Funktionen von PHP nicht nur benutzen, um die Grösse von JPEG, GIF, PNG und SWF-Bild-Dateien zu ermitteln, sondern auch - sofern Sie die GD-Bibliothek (verfügbar unter http://www.boutell.com/gd/) eingebunden haben - Grafiken bzw. Bilder dynamisch (also zur Laufzeit ihres Skripts) erzeugen bzw. verändern.

Die mittels dieser Funktionen veränderbaren Grafik-Formate sind abhängig von der bei ihnen installierten GD-Version. Versionen vor gd-1.6 unterstützen das GIF- aber kein PNG-Format. Die Versionen danach unterstützen zwar das PNG- aber nicht mehr das GIF-Format.

Um JPEG-Grafiken lesen und schreiben zu können, benötigen Sie jpeg-6b (siehe unter ftp://ftp.uu.net/graphics/jpeg/). Anschließend müssen Sie gd neu compilieren mit `--with-jpeg-dir=/path/to/jpeg-6b`. Danach muss auch PHP mit der Option `--with-jpeg-dir=/path/to/jpeg-6b` neu compiliert werden.

Die Unterstützung von Type-1-Fonts können Sie mit der Installation von t1lib (siehe unterftp://sunsite.unc.edu/pub/Linux/libs/graphics/) einbinden. Beim Compilieren müssen Sie dann noch die Option `--with-t1lib[=dir]` angeben.

GetImageSize (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Ausmaße einer GIF-, JPEG-, PNG- oder SWF-Grafik-Datei

array **getimagesize** (string filename [, array imageinfo])

Mit **GetImageSize()** ermitteln Sie die Daten jeder GIF-, JPG-, PNG oder SWF-Datei. Es werden die Ausmaße in Abhängigkeit vom Grafik-Typ sowie die Höhe und Breite als *ein* String zurückgegeben. Dieser kann innerhalb eines regulären HTML `IMG`-Tags verwendet werden.

Es wird ein Array aus 4 Elementen zurück gegeben. Im Array-Index 0 steht die Breite. Index 1 enthält die Höhe, 2 ein Flag je nach Grafik-Typ (1 = GIF, 2 = JPG, 3 = PNG, SWF = 4) und Index 3 die richtige Zeichenkette im Format "height=xxx width=xxx" zur Verwendung im IMG-Tag von HTML.

Beispiel 1. Grafik-Ausmaße ermitteln per GetImageSize:

```
<?php $size = GetImageSize ("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

Mit dem optionalen *`imageinfo`*-Parameter können Sie weitere Informationen der Grafik-Datei auslesen. Z.Zt. unterstützt dies die verschiedenen JPG APP-Markierungen per assoziativem Array. Einige Programme nutzen diese Markierungen, um Text in diese Grafiken einzubinden. Gebräuchlich ist die Einbettung von IPTC http://www.iptc.org/ Informationen in die APP13-Marke. Sie können iptcparse() benutzen, um die binäre APP13-Marke in ein lesbares Format zu übertragen.

Beispiel 2. GetImageSize mit IPTC-Informationen

```
<?php
    $size = GetImageSize ("testimg.jpg",&$info);
    if (isset ($info["APP13"])) {
        $iptc = iptcparse ($info["APP13"]);
        var_dump ($iptc);
    }
?>
```

Anmerkung: Dieser Befehl funktioniert auch ohne GD-Bibliothek.

ImageArc (PHP 3, PHP 4 >= 4.0.0)

Zeichnen einer Teil-Ellipse

int **imagearc** (int im, int cx, int cy, int w, int h, int s, int e, int col)

ImageArc() zeichnet den Teil einer Ellipse mit Mittelpunkt bei *cx* und *cy* (oben links ist 0, 0) in dem durch den *im*-Handle definiertem Bild. *w* und *h* benennen die Breite und Höhe der Ellipse, *s* und *e* den Start- bzw. Endwinkel des Bogens in Grad. Mit *col* bestimmen Sie die Farbe der Ellipse.

ImageChar (PHP 3, PHP 4 >= 4.0.0)

Stellt ein Zeichen mit horizontaler Ausrichtung dar

int **imagechar** (int im, int font, int x, int y, string c, int col)

ImageChar() zeichnet das erste Zeichen (Charakter) des Strings *c* in das Bild, auf das per *id* gezeigt wird. Die linke obere Ecke des Zeichens wird durch *x* / *y* bestimmt. Dabei wird von den Koordinaten 0 / 0 als Zeichnungsanfang links-oben ausgegangen. Die Farbe des Zeichens wird mit *col* bestimmt. Ist *font* als 1, 2, 3, 4 oder 5 definiert, wird ein eingebauter Font genutzt (je grösser die Nummer, desto grösser der Font).

Beachten Sie auch imageloadfont().

ImageCharUp (PHP 3, PHP 4 >= 4.0.0)

Zeichnet einen vertikal ausgerichteten Charakter

int **imagecharup** (int im, int font, int x, int y, string c, int col)

ImageCharUp() zeichnet das erste Zeichen (Charakter) des Strings *c* in das Bild, auf das per *id* gezeigt wird. Die Koordinaten des Zeichens werden durch *x* / *y* bestimmt. Dabei wird von den Koordinaten 0 / 0 als Zeichnungsanfang links-oben ausgegangen. Die Farbe des Zeichens wird mit *col* bestimmt. Ist *font* als 1, 2, 3, 4 oder 5 definiert, wird ein eingebauter Font genutzt (je grösser die Nummer, desto grösser der Font).

Beachten Sie auch imageloadfont().

ImageColorAllocate (PHP 3, PHP 4 >= 4.0.0)

Bestimmt die Farbe einer Grafik

int **imagecolorallocate** (int im, int red, int green, int blue)

ImageColorAllocate() gibt eine Farb-ID, die durch die angegebenen RGB-Werte bestimmt wird, zurück. Der Parameter *im* wird ihnen von der imagecreate()-Funktion zurück gegeben. **ImageColorAllocate()** muss für jede Farbe aufgerufen werden, die innerhalb der Grafik *im* verwendet werden soll.

```
$weiss   = ImageColorAllocate ($im, 255, 255, 255);
$schwarz = ImageColorAllocate ($im, 0, 0, 0);
```

ImageColorAt (PHP 3, PHP 4 >= 4.0.0)

Ermittelt den Farbwert eines Bildpunktes

int **imagecolorat** (int im, int x, int y)

Ermittelt den Farb-Wert eines Pixels an den Koordinaten *x* / *y* der mit *im* bestimmten Grafik.

Beachten Sie auch imagecolorset() und imagecolorsforindex().

ImageColorClosest (PHP 3, PHP 4 >= 4.0.0)

Ermittelt den Farbwert-Index, der den angegebenen Farben am nächsten liegt

int **imagecolorclosest** (int im, int red, int green, int blue)

Gibt den Index der Farbwert-Palette des Bildes *im* zurück, der den angegebenen RGB-Werten am nächsten kommt.

Die "Entfernung" zwischen der angeforderten Farbe und jedem Farb-Paletten-Wert wird berechnet, als ob die angegebenen RGB-Werte Punkte in einem dreidimensionalen Raum darstellen.

Siehe auch imagecolorexact().

ImageColorDeAllocate (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Löscht eine Farbdefinition

int **imagecolordeallocate** (int im, int index)

Die Funktion **ImageColorDeAllocate()** löscht eine vorher über **ImageColorAllocate()** ermittelte Farbe.

```
$weiss = ImageColorAllocate($im, 255, 255, 255);
ImageColorDeAllocate($im, $weiss);
```

ImageColorExact (PHP 3, PHP 4 >= 4.0.0)

Ermittelt den Index-Wert der angegebenen Farbe

int **imagecolorexact** (int im, int red, int green, int blue)

Gibt den Index der angegebenen Farbe in der Farb-Palette des Bildes *im* zurück.

Existiert die Farbe in der Bild-Farb-Palette nicht, wird -1 zurück gegeben.

Siehe auch imagecolorclosest().

ImageColorResolve (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Ermittelt den Index-Wert der angegebenen Farbe oder die nächst mögliche Alternative dazu

int **imagecolorresolve** (int im, int red, int green, int blue)

Diese Funktion ermittelt garantiert einen Farb-Index zu der angegebenen Farbe, nämlich entweder den exakten oder den nächst möglichen Wert.

Siehe auch imagecolorclosest() und imagecolorexact().

ImageColorSet (PHP 3, PHP 4 >= 4.0.0)

Setzt die Farbe für den angegebenen Paletten-Index

bool **imagecolorset** (int im, int index, int red, int green, int blue)

Hierdurch setzen Sie den Farbwert der Palette auf die angegebene Farbe. Dies ist nützlich bei Aktionen wie "Fülle Farbe X mit Farbe Y" (Flood-Fill-Effekt) ohne den sonst notwendigen Aufwand für solche Aktionen betreiben zu müssen.

Siehe auch imagecolorat().

ImageColorsForIndex (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Farbwerte einer angegebenen Farb-Palette

array **imagecolorsforindex** (int im, int index)

Gibt ein assoziatives Array mit Rot-, Grün- und Blau-Werten zurück, die die Werte des Farb-Paletten-Index von *im* darstellen.

Siehe auch imagecolorat() und imagecolorexact().

ImageColorsTotal (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Anzahl der definierten Farben eines Bildes

int **imagecolorstotal** (int im)

Ergibt die Anzahl der Farben der Palette des angegebenen Bildes *im*.

Siehe auch imagecolorat() und imagecolorsforindex().

ImageColorTransparent (PHP 3, PHP 4 >= 4.0.0)

Definiert eine Farbe als transparent

int **imagecolortransparent** (int im [, int col])

ImageColorTransparent() setzt die transparente Farbe des Bildes *im* auf die Farbe *col*. *Im* ist der Bild-Pointer, der von **ImageCreate()** zurück gegeben wird und *col* ist ein Zeiger auf die Farbe, die von **ImageColorAllocate()** ermittelt wird.

Es wird ein Zeiger auf die neue - jetzt transparente - Farbe (oder die aktuelle, falls keine Farbe angegeben wurde) zurück gegeben.

ImageCopy (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Kopiert einen Bildausschnitt

int **ImageCopy** (int dst_im, int src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h)

Kopiert einen Teil des Quellbildes *src_im* nach *dst_im*. Der zu kopierende Teil wird angegeben über die x/y-Koordinaten *src_x* und *src_y*. Die Breite wird in *src_w* und die Höhe in *src_h* angegeben. Der Ausschnitt wird an die x/y-Koordinaten *dst_x* und *dst_y* kopiert.

ImageCopyResized (PHP 3, PHP 4 >= 4.0.0)

Kopieren und Ändern der Grösse eines Bild-Teiles

int **imagecopyresized** (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)

ImageCopyResized() kopiert einen rechteckigen Ausschnitt eines Bildes in ein anderes. *Dst_im* gibt das Zielbild, *src_im* die Bildquelle an. Unterscheiden sich die Angaben der Quelle und des Ziels bezogen auf die Koordinaten, Höhe oder Breite, wird das Teilbild entsprechend gedehnt oder geschrumpft. Die Koordinaten beziehen sich auf die linke obere Ecke. Sie können hiermit auch Teilbilder innerhalb ein- und desselben Bildes kopieren, sofern *dst_im* und *src_im* gleich sind. Falls sich dabei aber die kopierten Teile überlappen, führt das zu unvorhersehbaren Ergebnissen.

ImageCreate (PHP 3, PHP 4 >= 4.0.0)

Erzeugt ein neues Bild

int **imagecreate** (int x_size, int y_size)

ImageCreate() gibt den Zeiger auf ein neues Bild zurück. Dieser Zeiger stellt ein leeres Bild mit der Größe *x_size* und *y_size* dar.

Beispiel 1. Erzeugung eines neuen GD-Bild-Streams und Ausgabe des Bildes:

```
<?php
header ("Content-type: image/png");
$im = @ImageCreate (50, 100)
      or die ("Kann keinen neuen GD-Bild-Stream erzeugen");
$background_color = ImageColorAllocate ($im, 255, 255, 255);
```

```
$text_color = ImageColorAllocate ($im, 233, 14, 91);
ImageString ($im, 1, 5, 5, "Ein Test-String", $text_color);
ImagePNG ($im);
?>
```

ImageCreateFromGIF (PHP 3, PHP 4 >= 4.0.0)

Erzeugt ein neues Bild im GIF-Format, welches aus einer Datei oder von einer URL gelesen wird

int **imagecreatefromgif** (string filename)

ImageCreateFromGIF() gibt den Bezeichner auf ein Bild zurück, der das aus der angegebenen Datei eingelesene Bild darstellt.

Im Fehlerfall gibt **ImageCreateFromGIF()** eine leere Zeichenkette zurück. Zudem wird eine Fehlermeldung erzeugt, die im Browser als Verbindungsabbruch dargestellt wird. Zum besseren Debuggen wird das folgende Beispiel einen GIF-Fehler erzeugen:

Beispiel 1. Beispiel, um die Handhabung eines Fehlers bei der Bilderzeugung zu sehen (Dank an vic@zymsys.com):

```
function LoadGif ($imgname) {
    $im = @ImageCreateFromGIF ($imgname); /* Versuch, Datei zu öffnen */
    if (!$im) {                           /* Prüfen, ob fehlgeschlagen */
        $im = ImageCreate (150, 30);      /* Erzeugen eines leeren Bildes */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Ausgabe einer Fehlermeldung */
        ImageString($im, 1, 5, 5, "Fehler beim Öffnen von: $imgname", $tc);
    }
    return $im;
}
```

Anmerkung: ACHTUNG: Ab Version 1.6 wurde die komplette GIF-Unterstützung aus der GD-Bibliothek entfernt. Diese Funktion ist also nur verfügbar, wenn Sie eine Version vor 1.6 verwenden.

ImageCreateFromJPEG (PHP 3>= 3.0.16, PHP 4 >= 4.0.0)

Erzeugt ein neues Bild im JPEG-Format, welches aus einer Datei oder von einer URL gelesen wird

int **imagecreatefromjpeg** (string filename)

ImageCreateFromJPEG() gibt den Bezeichner auf ein Bild zurück, der das aus der angegebenen Datei eingelesene Bild darstellt.

Im Fehlerfall gibt **ImageCreateFromJPEG()** eine leere Zeichenkette zurück. Zudem wird eine Fehlermeldung erzeugt, die im Browser als Verbindungsabbruch dargestellt wird. Zum besseren Debuggen wird das folgende Beispiel einen JPEG-Fehler erzeugen:

Beispiel 1. Beispiel, um die Handhabung eines Fehlers bei der Bilderzeugung zu sehen (Dank an vic@zymsys.com):

```
function LoadJpeg ($imgname) {
    $im = @ImageCreateFromJPEG ($imgname); /* Versuch, Datei zu öffnen */
    if (!$im) {                            /* Prüfen, ob fehlgeschlagen */
        $im = ImageCreate (150, 30);       /* Erzeugen eines leeren Bildes */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Ausgabe einer Fehlermeldung */
        ImageString($im, 1, 5, 5, "Fehler beim Öffnen von: $imgname", $tc);
    }
    return $im;
}
```

ImageCreateFromPNG (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Erzeugt ein neues Bild im PNG-Format, welches aus einer Datei oder von einer URL gelesen wird

int **imagecreatefrompng** (string filename)

ImageCreateFromPNG() gibt den Bezeichner auf ein Bild zurück, der das aus der angegebenen Datei eingelesene Bild darstellt.

Im Fehlerfall gibt **ImageCreateFromPNG()** eine leere Zeichenkette zurück. Zudem wird eine Fehlermeldung erzeugt, die im Browser als Verbindungsabbruch dargestellt wird. Zum besseren Debuggen wird das folgende Beispiel einen JPEG-Fehler erzeugen:

Beispiel 1. Beispiel, um die Handhabung eines Fehlers bei der Bilderzeugung zu sehen (Dank an vic@zymsys.com):

```
function Loadpng ($imgname) {
    $im = @ImageCreateFromPNG ($imgname); /* Versuch, Datei zu öffnen */
    if (!$im) {                           /* Prüfen, ob fehlgeschlagen */
        $im = ImageCreate (150, 30);      /* Erzeugen eines leeren Bildes */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Ausgabe einer Fehlermeldung */
        ImageString($im, 1, 5, 5, "Fehler beim Öffnen von: $imgname", $tc);
    }
    return $im;
}
```

ImageDashedLine (PHP 3, PHP 4 >= 4.0.0)

Zeichnen einer gestrichelten Linie

int **imagedashedline** (int im, int x1, int y1, int x2, int y2, int col)

ImageDashedLine() zeichnet eine gestrichelte Linie, beginnend am Punkt `x1`, `y1` bis zum Punkt `x2`, `y2` (oben links ist 0, 0) in das Bild image `im` mit der Farbe `col`.

Siehe auch imageline().

ImageDestroy (PHP 3, PHP 4 >= 4.0.0)

Löscht ein Bild

int **imagedestroy** (int im)

ImageDestroy() gibt den durch das Bild `im` belegten Speicher wieder frei. `Im` ist der Bezeichner, der ihnen beim Aufruf der Funktion **ImageCreate()** zurück gegeben worden ist.

ImageFill (PHP 3, PHP 4 >= 4.0.0)

Füllen mit Farbe ("flood fill")

int **imagefill** (int im, int x, int y, int col)

ImageFill() bewirkt das Füllen eines Bildes beginnend bei der Koordinate `x`, `y` (oben links ist 0, 0) mit der Farbe `col` im Bild `im`.

ImageFilledPolygon (PHP 3, PHP 4 >= 4.0.0)

Zeichnet ein gefülltes Vieleck (Polygon)

int **imagefilledpolygon** (int im, array points, int num_points, int col)

ImageFilledPolygon() erzeugt ein Vieleck im Bild `im`, gefüllt mit der Farbe `col`. `Points` ist ein PHP-Array, das die Eckpunkte des Vielecks enthält. Points[0] ist die X-Koordinate (x0), points[1] die Y-Koordinate (y0) des ersten Eckunktes. Points[2] ist x1, points[3] = y1 usw. `Num_points` enthält die Anzahl der Punkte.

ImageFilledRectangle (PHP 3, PHP 4 >= 4.0.0)

Zeichnet ein gefülltes Rechteck

int **imagefilledrectangle** (int im, int x1, int y1, int x2, int y2, int col)

ImageFilledRectangle() erzeugt ein mit der Farbe **col()** gefülltes Rechteck innerhalb des Bildes `im`. Die obere linke Eck-Koordinate wird mittels `x1` und `y1`, die untere rechte Eck-Koordinate mittels `x2` und `y2` definiert. 0, 0 ist dabei die linke obere Ecke des Bildes `im`.

ImageFillToBorder (PHP 3, PHP 4 >= 4.0.0)

Flächen-Farbfüllung ("flood fill") mit einer angegebenen Farbe

int **imagefilltoborder** (int im, int x, int y, int border, int col)

ImageFillToBorder() bewirkt eine Farb-Füllung in einem Bereich, dessen Grenze durch die Farbe in `border` definiert wird. Der Startpunkt wird durch `x` und `y` angegeben (oben links ist 0, 0). Der Bereich wird gefüllt mit der Farbe `col`.

ImageFontHeight (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Font-Höhe

int **imagefontheight** (int font)

Ermittelt die Höhe eines Zeichens (in Pixel) im angegebenen Font.

Siehe auch imagefontwidth() und imageloadfont().

ImageFontWidth (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Font-Breite

int **imagefontwidth** (int font)

Gibt die Zeichenbreite des angegebenen Fonts in Pixeln zurück.

Siehe auch imagefontheight() und imageloadfont().

ImageGammaCorrect (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Anwendung einer Gamma-Korrektur auf ein GD-Bild

int **imagegammacorrect** (int im, double inputgamma, double outputgamma)

Die Funktion **ImageGammaCorrect()** bewirkt die Anwendung einer Gamma-Korrektur auf das mittels Bezeichner `im` angegebene GD-Bild. Anzugeben ist der Eingabe-Gammawert `inputgamma` und der Ausgabe-Gammawert `outputgamma`.

ImageGIF (PHP 3, PHP 4 >= 4.0.0)

Ausgabe eines Bildes an den Browser oder in eine Datei

int **imagegif** (int im [, string filename])

ImageGIF() erzeugt eine Datei im Format GIF aus *im*. Falls Sie [filename] angegeben haben, wird die Ausgabe in diese Datei geschrieben, ansonsten wird die Ausgabe direkt an den Browser geleitet. *Im* ist das Ergebnis der imagecreate()-Funktion.

Das Format wird GIF87a sein, es sei denn, Sie haben das Bild per **ImageColorTransparent()** als transparent formatiert. In diesem Fall wird das Bild im GIF89a-Format ausgegeben.

Die Angabe des Dateinamens ist optional. Falls er fehlt, werden die Bilddaten direkt ausgegeben. Durch Verwendung der Funktion header() mit der "content-type"-Angabe "image/gif" können Sie PHP-Skripte erstellen, welche die GIF-Ausgabe direkt vornehmen.

Anmerkung: ACHTUNG: Ab Version 1.6 wurde die GIF-Unterstützung komplett aus der GD-Bibliothek entfernt. Diese Funktion ist also nur verfügbar, wenn Sie eine Version vor 1.6 verwenden.

Um trotzdem portable PHP-Anwendungen schreiben zu können, kann mit folgendem Code-Stück automatisch geprüft werden, welches Bildformat in der mitgebundenen GD-Bibliothek nutzbar ist. Ersetzen Sie hierfür die Codesequenz `Header("Content-type: image/gif"); ImageGif($im);` durch die flexiblere Folge:

```
<?php
  if (function_exists("imagegif")) {
    Header("Content-type: image/gif");
    ImageGif($im);
  }
  elseif (function_exists("imagejpeg")) {
    Header("Content-type: image/jpeg");
    ImageJpeg($im, "", 0.5);
  }
  elseif (function_exists("imagepng")) {
    Header("Content-type: image/png");
    ImagePng($im);
  }
  else
    die("Keine Grafik-Unterstützung vorhanden");
?>
```

Anmerkung: In PHP 4 ab Version 4.0.2 haben Sie alternativ zum Test per function_exists() auch die Möglichkeit, mit der imagetypes()-Funktion die unterstützten Bildformate abzutesten.

```
if (ImageTypes() & IMG_GIF) {
    Header("Content-type: image/gif");
    ImageGif($im);
}
elseif (ImageTypes() & IMG_JPG) {
... etc.
```

Siehe auch imagepng(), imagejpeg(), imagetypes().

ImageJPEG (PHP 3>= 3.0.16, PHP 4 >= 4.0.0)

Ausgabe des Bildes im Browser oder als Datei

int **imagejpeg** (int im [, string filename [, int quality]])

ImageJPEG() erzeugt die Datei *filename* im JPEG-Format aus dem Zeiger auf *im*. Das Argument *im* wurde ihnen vorher durch den Aufruf der Funktion **ImageCreate()** mitgeteilt.

Die Angabe des Dateinamens *filename* ist optional. Fehlt er, werden die erzeugten Bilddaten direkt ausgegeben. Um die Angabe des Dateinamen zu umgehen, dennoch aber den Parameter als solchen zu nutzen, können Sie einen Leerstring (”) benutzen. Durch Verwendung der Funktion header() mit der "content-type"-Angabe "image/jpeg" können Sie PHP-Skripte erstellen, welche die JPEG-Ausgabe direkt vornehmen.

Anmerkung: Die JPEG-Unterstützung können Sie nur verwenden, wenn Sie PHP mit einer GD-Bibliothek ab Version 1.8 kompiliert haben.

Siehe auch imagepng(), imagegif(), imagetypes().

ImagePNG (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ausgabe eines Bildes an den Browser oder in eine Datei

int **imagepng** (int im [, string filename])

ImagePNG() erzeugt eine Datei im Format PNG aus *im*. Falls Sie [filename] angegeben haben, wird die Ausgabe in diese Datei geschrieben, ansonsten wird die Ausgabe direkt an den Browser geleitet. *Im* ist das Ergebnis der imagecreate()-Funktion.

```
<?php
$im = ImageCreateFromPNG("test.png");
ImagePNG($im);
?>
```

Die Angabe des Dateinamens ist optional. Fehlt er, werden die Bilddaten direkt ausgegeben. Durch Verwendung der Funktion header() mit der "content-type"-Angabe "image/png" können Sie PHP-Skripte erstellen, welche die PNG-Ausgabe direkt vornehmen.

Anmerkung: ACHTUNG: Die PNG-Unterstützung ersetzt die GIF-Funktionen ab der GD-Bibliothek Version 1.6. Diese Funktion ist also nur verfügbar, wenn Sie eine Version ab 1.6 verwenden.

Siehe auch imagegif(), imagejpeg(), imagetypes().

ImageInterlace (PHP 3, PHP 4 >= 4.0.0)

Schaltet die Interlaced-Darstellung eines Bildes an oder aus

int **imageinterlace** (int im [, int interlace])

Mit der Funktion **imageInterlace()** setzen Sie das Bit für die interlaced-Darstellung eines Bildes auf An (1) oder Aus (0).

Als Rückgabewert erhalten Sie die aktuelle Einstellung des Interlaced-Modus.

ImageLine (PHP 3, PHP 4 >= 4.0.0)

Zeichnen einer Linie

int **imageline** (int im, int x1, int y1, int x2, int y2, int col)

ImageLine() zeichnet eine Linie von der *x1* / *y1* Koordinate bis zur Kordinate *x2* / *y2* (die Koordinate 0 / 0 ist die obere linke Ecke von *im*). Die Linienfarbe wird durch *col* festgelegt.

Siehe auch imagecreate() und imagecolorallocate().

ImageLoadFont (PHP 3, PHP 4 >= 4.0.0)

Lädt einen neuen Font

int **imageloadfont** (string file)

ImageLoadFont() lädt einen vom Benutzer definierten Bitmap-Font und gibt einen Zeiger auf diesen zurück. Dessen Wert ist immer grösser als 5, um Konflikte mit den eingebauten Fonts zu vermeiden.

Das Font-Datei-Format unterliegt binären Abhängigkeiten und ist von Rechner-Typ zu Rechner-Typ unterschiedlich. Das bedeutet, dass Sie die Font-Dateien immer auf dem Rechner-Typ erzeugen müssen, auf dem diese Verwendung finden.

Tabelle 1. Font-Datei-Format

Byte-Position	C Daten-Typ	Beschreibung
Byte 0-3	int	Anzahl der Zeichen des Font
Byte 4-7	int	Wert des ersten Zeichens des Fonts (meist 32 für Leerzeichen)

Byte-Position	C Daten-Typ	Beschreibung
Byte 8-11	int	Pixel-Breite jedes Zeichens
Byte 12-15	int	Pixel-Höhe jedes Zeichens
Byte 16-	Zeichen	Zeichen-Daten als Array, ein Byte je Pixel für jedes Zeichen; in der Summe also (Anzahl-Zeichen * Breite * Höhe)-Bytes.

Siehe auch imagefontwidth() und imagefontheight().

ImagePolygon (PHP 3, PHP 4 >= 4.0.0)

Zeichnen eines Vielecks (Polygon)

int **imagepolygon** (int im, array points, int num_points, int col)

ImagePolygon() erzeugt ein Vieleck innerhalb von *im*. *Points* ist ein PHP-Array, das die Werte für die Eckpunkte des Polygons enthält, z.B. Points[0] = x0, Points[1] = y0, Points[2] = x1, Points[3] = y1 usw. *Num_points* enthält die Gesamtzahl an Points (Points[n]).

Siehe auch imagecreate() und imagefilledpolygon().

ImagePSBBox (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Ermittelt die Ausmaße des Rechtecks, das für die Ausgabe eines Textes unter Verwendung eines PostScript-Fonts (Typ 1) notwendig ist.

array **imagepsbbox** (string text, int font, int size [, int space [, int tightness [, float angle]]])

Size ist die Größe in Pixel.

Space ermöglicht ihnen, den vorgegebenen Wert für den Zeichen-Zwischenraum eines Fonts anzugeben. Dieser Wert wird zum Normal-Wert hinzu gezählt und kann auch negativ sein.

Tightness ermöglicht ihnen die Kontrolle über die Breite eines Leerzeichens. Der angegebene Wert wird zur normalen Zeichenbreite hinzu gezählt und kann auch negativ sein.

Angle (Winkel) wird in Grad angegeben und beinhaltet den Drehungswinkel des auszugebenden Textes.

Die Parameter *space* und *tightness* haben als Einheit die Zeichenbreite, wobei der Wert 1 gleich 1/1000stel der Breite des Zeichens 'm' entspricht (Maß 'em').

Die Parameter *space*, *tightness* und *angle* sind optional.

Das umgebende Rechteck wird berechnet unter Verwendung der für die Zeichen-Abmessungen verfügbaren Informationen und unterscheidet sich immer etwas von den aktuellen Raster-Werten des Textes. Ist der Winkel (angle) mit 0 angegeben, können Sie davon ausgehen, dass der Text in jede Richtung immer 1 Pixel mehr braucht als sonst.

Diese Funktion gibt ein Array zurück, welches die folgenden Elemente enthält:

0	untere linke x-Koordinate
1	untere linke y-Koordinate
2	obere rechte x-Koordinate
3	obere rechte y-Koordinate

Siehe auch imagepstext().

ImagePSEncodeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Ändert die Vektor-Beschreibung eines Fonts

int **imagepsencodefont** (string encodingfile)

Lädt einen Zeichen-Beschreibungs-Vektor aus einer Datei und verändert die Vektor-Beschreibung des Fonts. Da ein PS-Font meist Lücken oberhalb des 127. Zeichens aufweist, werden Sie dies oft anpassen wollen, sofern Sie eine andere Sprache als Englisch benutzen. Das genaue Dateiformat ist in der Dokumentation zu T1Lib beschrieben. T1Lib wird mit zwei gebrauchsfertigen Beschreibungsdateien (IsoLatin1.enc und IsoLatin2.enc) ausgeliefert.

Sollten Sie diese Funktion ständig benutzen, ist es besser, diese Beschreibungs-Datei innerhalb der Konfigurations-Datei bei ps.default_encoding anzugeben. Dadurch wird immer die richtige Datei geladen und die Zeichen haben damit die richtige Beschreibung.

ImagePsExtendFont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Vergrößert oder komprimiert einen Font

bool **imagepsextendfont** (int font_index, double extend)

Der mit `font_index` angegebene Font wird gedehnt (`extend` ist größer als der Font) oder verkleinert (`extend` ist kleiner als der Font).

ImagePSFreeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Gibt den durch einen Typ 1 PostScript-Font belegten Speicher wieder frei

void **imagepsfreefont** (int fontindex)

Siehe auch imagepsloadfont().

ImagePSLoadFont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Lädt einen Typ 1 PostScript-Font aus einer Datei

int **imagepsloadfont** (string filename)

Wenn alles in Ordnung ist, wird ein gültiger Zeiger (Index) zurück gegeben, auf den Sie immer wieder zurück greifen können; ansonsten wird `FALSE` zurück und eine Fehlermeldung ausgegeben.

Siehe auch imagepsfreefont().

ImagePsSlantFont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Setzt einen Font schräg

bool **imagepsslantfont** (int font_index, double slant)

Der mit *font_index* angegebene Font wird um den Faktor *slant* "verschrägt" (italic!).

ImagePSText (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Ausgabe eines Textes auf einem Bild unter Verwendung von Typ 1 PostScript-Fonts

array **imagepstext** (int image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]])

Size wird in Pixel ausgedrückt.

Foreground ist die Text-Ausgabefarbe (Vordergrund). *Background* ist die Farbe, zu der ein Farbübergang versucht werden soll. Dabei wird kein Pixel mit der Farbe *Background* ausgegeben, so dass das Bild im Hintergrund keine festen Farben benötigt.

Die Koordinaten *x* und *y* definieren den Ursprung (bzw. den Referenz-Punkt) des ersten Zeichens (ungefähr die linke untere Ecke). Dies ist anders als bei der **ImageString()**-Funktion, wo *x* und *y* die obere linke Ecke des ersten Zeichens angeben. Lesen Sie in der Dokumentation zu PostScript und seines Maßsystems nach, falls Sie die Funktionsweise nicht verstehen.

Space ermöglicht ihnen, die Vorgabe des Wertes für den Zeichen-Zwischenraum zu verändern. Der angegebene Wert wird addiert und kann auch negativ sein.

Tightness ermöglicht die Kontrolle über die Breite des Leerzeichens. Der Wert wird addiert und kann auch negativ sein.

Angle wird in Grad angegeben und hat die entsprechende Drehung des Ausgabetextes zur Folge.

Antialias_steps erlaubt die Kontrolle über die Anzahl der Farben, mit der der Farbübergang vom Text zum Hintergrundbild vorgenommen werden soll. Möglich sind Werte von 4 bis 16. Der höhere Wert wird empfohlen bei Textgrössen kleiner als 20, wo der Effekt bei der Textqualität deutlich sichtbar ist. Bei grösseren Werten als 20 werden 4 Farben empfohlen. Das ist weniger rechenintensiv.

Die Parameter *space* und *tightness* werden in 1/1000-stel der Breite des Buchstabens "m" (Einheit 'em') angegeben.

Die Parameter *space*, *tightness*, *angle* und *antialias* sind optional.

Diese Funktion gibt ein Array mit den folgenden Werten zurück:

0	untere linke X-Koordinate

1	untere linke Y-Koordinate
2	obere rechte X-Koordinate
3	obere rechte Y-Koordinate

Siehe auch imagepsbbox().

ImageRectangle (PHP 3, PHP 4 >= 4.0.0)

Zeichnet ein Rechteck

int **imagerectangle** (int im, int x1, int y1, int x2, int y2, int col)

ImageRectangle() erzeugt ein Rechteck mit der Linienfarbe *col*, beginnend bei der oberen linken Koordinate X1 und Y1. Das Ende liegt bei der unteren rechten Koordinate X2 und Y2. Dabei ist 0, 0 die linke obere Ecke des Bildes *im*.

ImageSetPixel (PHP 3, PHP 4 >= 4.0.0)

Setzt ein einzelnes Pixel

int **imagesetpixel** (int im, int x, int y, int col)

ImageSetPixel() zeichnet ein Pixel an der Koordinate *x* / *y*. Dabei ist die Koordinate 0, 0 die linke obere Ecke des Bildes *im*. Die Pixelfarbe wird durch *col* bestimmt.

Siehe auch imagecreate() und imagecolorallocate().

ImageString (PHP 3, PHP 4 >= 4.0.0)

Zeichnet einen horizontalen String

int **imagestring** (int im, int font, int x, int y, string s, int col)

ImageString() gibt den String *s* in dem durch den Parameter *im* bezeichneten Bild an den Koordinaten *x* und *y* aus. Die Koordinaten 0, 0 geben die linke obere Ecke des Bildes *im* an. Der Font erscheint in der Farbe *col*. Hat *font* den Wert 1, 2, 3, 4 oder 5 wird ein interner Font benutzt.

Siehe auch imageloadfont().

ImageStringUp (PHP 3, PHP 4 >= 4.0.0)

Zeichnet einen vertikalen String

int **imagestringup** (int im, int font, int x, int y, string s, int col)

ImageStringUp() gibt den String `s` vertikal innerhalb des Bildes `im` an den Koordinaten `x` und `y` aus. Die linke obere Ecke von `im` ist dabei 0, 0. Der Text wird in der Farbe `col` ausgegeben. Hat `font` den Wert 1, 2, 3, 4 oder 5 wird ein interner Font benutzt.

Siehe auch imageloadfont().

ImageSX (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Bild-Breite

int **imagesx** (int im)

ImageSX() ermittelt die Breite des Bildes, das durch `im` referenziert wird.

Siehe auch imagecreate() und imagesy().

ImageSY (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Bild-Höhe

int **imagesy** (int im)

ImageSY() ermittelt die Höhe des Bildes, das durch `im` referenziert wird.

Siehe auch imagecreate() und imagesx().

ImageTTFBBox (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Ermittelt die Rahmenmaße für die Ausgabe eines Textes im True-Type-Format

array **imagettfbbox** (int size, int angle, string fontfile, string text)

Diese Funktion errechnet die Grösse des Bereichs für eine True-Type-Textausgabe.

`text`

Der zu messende Text

`size`

Die Font-Größe

`fontfile`

Der Name der TTF-Datei (kann auch eine URL sein)

`angle`

Der Winkel, in dem `text` ausgegeben wird

ImageTTFBBox() gibt ein Array mit acht Werten zurück. Diese enthalten die vier Punkte, durch die der Ausgabebereich markiert wird:

0	untere linke Ecke, X-Position
1	untere linke Ecke, Y-Position
2	untere rechte Ecke, X-Position
3	untere rechte Ecke, Y-Position
4	obere rechte Ecke, X-Position
5	obere rechte Ecke, Y-Position
6	obere linke Ecke, X-Position
7	obere linke Ecke, Y-Position

Die Punkte sind relativ zum *text*. Ohne Rücksicht auf den Winkel ist oben-links also die obere linke Ecke, wobei die Textausgabe horizontal gesehen wird.

Diese Funktion erfordert sowohl die GD- als auch die FreeType (http://www.freetype.org/)-Bibliothek.

Siehe auch imagettftext().

ImageTTFText (PHP 3, PHP 4 >= 4.0.0)

Erzeugt TTF-Text im Bild

array **imagettftext** (int im, int size, int angle, int x, int y, int col, string fontfile, string text)

ImageTTFText() zeichnet den Text `text` in das per `im` angegebene Bild, beginnend bei den Koordinaten `x` und `y` (oben links ist 0, 0) mit einem Ausgabewinkel von `angle` in der Farbe `col` unter Verwendung der TTF-Datei, die mittels `fontfile` benannt wurde.

Die per `x` und `y` angegebenen Koordinaten definieren den Startpunkt des ersten Zeichens (in etwa die linke untere Ecke). Das ist anders als bei der **ImageString()**-Funktion, bei der x und y die obere rechte Ecke des ersten Zeichens darstellen.

`Angle` ist in Grad anzugeben. 0 ist eine normale Ausrichtung von links nach rechts (von 9 nach 3 Uhr). Höhere Werte entsprechen einer immer weiteren Drehung entgegen dem Uhrzeiger-Sinn. Eine Drehung von 90 Grad entspricht also einer Text-Ausrichtung von unten nach oben (von 6 nach 12 Uhr).

`Fontfile` ist der Pfad zur TTF-Datei, die Sie verwenden wollen.

`Text` ist der auszugebende String, der auch UTF-8-Zeichenfolgen enthalten darf (z.B. {)

`Col` ist der Farbindex. Verwenden Sie den negativen Index-Wert, führt das zum Aufheben des Farbüberlaufs (Antialiasing).

ImageTTFText() gibt ein Array aus acht Elementen zurück, das die vier Ecken des Textausgabe-Bereichs enthält (wie **ImageTTFBBox()**). Die Reihenfolge ist dabei oben links, oben rechts, unten rechts und unten links. Die Punkte sind dabei relativ zum Text. Deshalb gibt oben links die linke Ecke bei horizontaler Sichtweise an.

Das Beispiel-Skript erzeugt eine schwarze GIF-Grafik von 400x30 Pixel mit dem weissen Text "Teste..." im TT-Font Arial.

Beispiel 1. ImageTTFText

```
<?php
```

```
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
              "Teste... Omega: &#937;");
ImageGif ($im);
ImageDestroy ($im);
?>
```

Diese Funktion erfordert sowohl die GD- als auch die FreeType (http://www.freetype.org/)-Bibliothek.

Siehe auch imagettfbbox().

ImageTypes (PHP 3 CVS only, PHP 4 >= 4.0.2)

Gibt die von der aktuell verwendeten PHP-Version unterstützten Grafik-Formate zurück

int **imagetypes** (void)

Der von dieser Funktion zurück gegebene Bit-Wert stellt die von der in der aktuellen PHP-Kompilation enthaltenen GD-Version unterstützten Grafik-Formate dar. Die folgenden Bit-Werte sind möglich: IMG_GIF, IMG_JPG, IMG_PNG oder IMG_WBMP. Um das Vorhanden sein der GD-Unterstützung zu testen, können Sie folgendes machen:

Beispiel 1. ImageTypes

```
<?php
if (ImageTypes() & IMG_PNG) {
    echo "PNG-Unterstützung ist vorhanden.";
}
?>
```

read_exif_data (PHP 4)

Liest die EXIF Header-Infos einer JPEG-Grafik

array **read_exif_data** (string filename)

Die Funktion **read_exif_data()** liest die EXIF Datei-Kopf-Informationen aus einer JPEG-Grafik-Datei. Es wird ein assoziatives Array in dem die Indizes die EXIF-Kopf-Bezeichnungen und die Werte die Inhalte dieser Kopf-Infos sind. EXIF-Infos sind häufig in JPEG-Grafiken enthalten, die mit einer Digital-Kamera aufgenommen wurden. Jeder Aufnehmende hat aber andere Vorstellungen über die Daten einer JPEG-Grafik. Diese Unterschiede können Sie mit dieser Funktion ermitteln.

Beispiel 1. read_exif_data

```
<?php
 $exif = read_exif_data ('p0001807.jpg');
 while(list($k,$v)=each($exif)) {
   echo "$k: $v<br>\n";
 }
?>

Ausgabe:
FileName: p0001807.jpg
FileDateTime: 929353056
FileSize: 378599
CameraMake: Eastman Kodak Company
CameraModel: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
DateTime: 1999:06:14 01:37:36
Height: 1024
Width: 1536
IsColor: 1
FlashUsed: 0
FocalLength:  8.0mm
RawFocalLength: 8
ExposureTime:  0.004 s (1/250)
RawExposureTime: 0.0040000001899898
ApertureFNumber: f/ 9.5
RawApertureFNumber: 9.5100002288818
FocusDistance: 16.66m
RawFocusDistance: 16.659999847412
Orientation: 1
ExifVersion: 0200
```

Anmerkung: Diese Funktion ist in PHP 4 nur verfügbar, wenn Sie es mit --enable-exif compiliert haben.

Diese Funktion erfordert nicht die GD-Library.

XLII. IMAP, POP3 und NNTP Funktionen

Dieses Paket baut auf auf der C-Client Bibliothek auf, die Sie unter der URL ftp://ftp.cac.washington.edu/imap/ erhalten. Kompilieren Sie diese und kopieren Sie `c-client/c-client.a` nach `/usr/local/lib/libc-client.a` sowie `c-client/rfc822.h`, `mail.h` und `linkage.h` nach `/usr/local/include` oder in ein anderes Verzeichnis in Ihrem Link- bzw. Include-Pfad. Anschließend konfigurieren und kompilieren Sie PHP mit der Option `--with-imap`.

Beachten Sie, daß diese Funktionen nicht auf das IMAP -Protokoll beschränkt sind, auch wenn der Name dies vermuten läßt. Die zugrundeliegende C-Client Bibliothek unterstützt auch POP3, NNTP und lokale Mailbox-Zugriffe.

Diese Dokumentation kann nicht alle Bereiche beschreiben, die von den enthaltenen Funktionen berührt werden. Weitergehende Informationen erhalten Sie in der Dokumentation der C-Client Bibliothek (Datei `docs/internal.txt` im Quellpaket der Bibliothek) sowie in folgenden RFC-Dokumenten:

- RFC821 (http://www.faqs.org/rfcs/rfc821.html) : Simple Mail Transfer Protocol (SMTP)
- RFC822 (http://www.faqs.org/rfcs/rfc822.html) : Standard for ARPA internet text messages
- RFC2060 (http://www.faqs.org/rfcs/rfc2060.html) : Internet Message Access Protocol (IMAP) Version 4rev1
- RFC1939 (http://www.faqs.org/rfcs/rfc1939.html) : Post Office Protocol Version 3 (POP3)
- RFC977 (http://www.faqs.org/rfcs/rfc977.html) : Network News Transfer Protocol (NNTP)
- RFC2076 (http://www.faqs.org/rfcs/rfc2076.html) : Common Internet Message Headers
- RFC2045 (http://www.faqs.org/rfcs/rfc2045.html) , RFC2046 (http://www.faqs.org/rfcs/rfc2046.html) , RFC2047 (http://www.faqs.org/rfcs/rfc2047.html) , RFC2048 (http://www.faqs.org/rfcs/rfc2048.html) & RFC2049 (http://www.faqs.org/rfcs/rfc2049.html) : Multipurpose Internet Mail Extensions (MIME)

Eine ausführliche Übersicht bietet auch das Buch Programming Internet Email (http://www.oreilly.com/catalog/progintemail/noframes.html) von David Wood.

imap_append (PHP 3, PHP 4 >= 4.0.0)

Fügt eine String-Nachricht an das angegebene Postfach an.

int **imap_append** (int imap_stream, string mbox, string message [, string flags])

Liefert `TRUE` bei Erfolg und `FALSE` bei Fehlern.

imap_append() fügt den String aus *`message`* als neue Nachricht an die durch *`mbox`* spezifizierte Mailbox an. Wenn der optionale *`flags`* spezifiziert wird, so werden die Flags der Nachricht in der Mailbox entsprechend gesetzt.

Beispiel 1. imap_append() Beispiel

```
$stream = imap_open ("{your.imap.host}INBOX.Drafts", "username", "password");

$check = imap_check ($stream);
print "Msg Count before append: " . $check->Nmsgs . "<br>\n";

imap_append( $stream, "{your.imap.host}INBOX.Drafts"
                    , "From: me@my.host\r\n"
                    . "To: you@your.host\r\n"
                    . "Subject: test\r\n"
                    . "\r\n"
                    . "Das ist ein Test, bitte ignorieren\r\n"
                    );

$check = imap_check ($stream);
print "Msg Count after append : " . $check->Nmsgs . "<br>\n";

imap_close ($stream);
```

Warnung

Der Cyrus IMAP Server erwartet zwingend "\r\n" als Zeilenende-Kennung, ein einfaches "\n" wird nicht richtig verarbeitet.

imap_base64 (PHP 3, PHP 4 >= 4.0.0)

Dekodiert BASE64-codierten Text

string **imap_base64** (string text)

Die **imap_base64()** Funktion decodiert BASE64 codierten Text (gemäß RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Abschnitt 6.8). Der decodierte Text wird als String zurückgegeben.

Siehe auch imap_binary().

imap_body (PHP 3, PHP 4 >= 4.0.0)

Liest den Körper einer Nachricht

string **imap_body** (int imap_stream, int msg_number [, int flags])

imap_body() liefert den Körper der Nachricht mit der Nummer *msg_number* im aktuellen Postfach. Die optionalen *flags* können eine Bitmaske enthalten, die sich aus folgenden Werten zusammensetzt:

- FT_UID - *msg_number* ist eine UID
- FT_PEEK - Das \Seen-Flag wird nicht beeinflußt.
- FT_INTERNAL - Die Nachricht wird im internen Format zurückgegeben eine CRLF-Umwandlung findet nicht statt.

imap_check (PHP 3, PHP 4 >= 4.0.0)

Prüft den Status des aktuelle Postfachs

object **imap_check** (int imap_stream)

Die **imap_check()** Funktion prüft den Status des aktuellen Postfach und liefert die entsprechenden Felder in einem Objekt mit folgenden Attributen zurück:

- Date - Datum der letzten Änderung des Postfach-Inhalts
- Driver - Zugriffsprotokoll: POP, IMAP, NNTP
- Mailbox - Name des Postfachs
- Nmsgs - Anzahl der Nachrichten im Postfach
- Recent - Anzahl der neuen Nachrichten im Postfach

Beispiel 1. imap_check() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    || die ("can't connect: " . imap_last_error());

$check = imap_check ($mbox);

if($check) {
    print "Date: "     . $check->Date    . "<br>\n" ;
    print "Driver: "   . $check->Driver  . "<br>\n" ;
    print "Mailbox: "  . $check->Mailbox . "<br>\n" ;
    print "Messages: " . $check->Nmsgs   . "<br>\n" ;
    print "Recent: "   . $check->Recent  . "<br>\n" ;
} else {
    print "imap_check() failed: " . imap_last_error() . "<br>\n";
}
```

```
imap_close ($mbox);
```

imap_close (PHP 3, PHP 4 >= 4.0.0)

Schließt eine IMAP-Verbindung

int **imap_close** (int imap_stream [, int flags])

Schließt eine IMAP-Verbindung. Mit dem optionalen *flag* CL_EXPUNGE kann ein Bereinigen (physikalisches Löschen aller zum Löschen markierten Nachrichten) ausgelöst werden.

imap_createmailbox (PHP 3, PHP 4 >= 4.0.0)

Erzeugt ein neues Postfach

int **imap_createmailbox** (int imap_stream, string mbox)

imap_createmailbox() legt ein über den Parameter *mbox* spezifiziertes neues Postfach an.

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

Siehe auch imap_renamemailbox() and imap_deletemailbox() sowie imap_open() für das Format des Parameters *mbox*.

Beispiel 1. imap_createmailbox() Beispiel

```
$mbox = imap_open ("{your.imap.host}", "username", "password", OP_HALFOPEN)
     || die ("can't connect: ".imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode ("phpnewböx");

$newname = $name1;

echo "Newname will be '$name1'<br>\n";

# we will now create a new mailbox "phptestbox" in your inbox folder,
# check its status after creation and finaly remove it to restore
# your inbox to its initial state

if (@imap_createmailbox ($mbox,imap_utf7_encode ("{your.imap.host}INBOX.$newname"))) {
    $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
    if($status) {
        print("your new mailbox '$name1' has the following status:<br>\n");
        print("Messages:   ". $status->messages   )."<br>\n";
        print("Recent:     ". $status->recent     )."<br>\n";
        print("Unseen:     ". $status->unseen     )."<br>\n";
        print("UIDnext:    ". $status->uidnext    )."<br>\n";
        print("UIDvalidity:". $status->uidvalidity)."<br>\n";
```

```
            if (imap_renamemailbox ($mbox,"{your.imap.host}INBOX.$newname", "{your.imap.host
                echo "renamed new mailbox from '$name1' to '$name2'<br>\n";
                $newname=$name2;
            } else {
                print "imap_renamemailbox on new mailbox failed: ".imap_last_error ()."<br>\n
            }
        } else {
                print "imap_status on new mailbox failed: ".imap_last_error()."<br>\n";
        }
        if (@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
            print "new mailbox removed to restore initial state<br>\n";
        } else {
            print  "imap_deletemailbox on new mailbox failed: ".implode ("<br>\n", imap_error
        }

    } else {

        print "could not create new mailbox: ".implode ("<br>\n",imap_errors())."<br>\n";
    }

    imap_close($mbox);
```

imap_delete (PHP 3, PHP 4 >= 4.0.0)

Merkt eine Nachricht des aktuellen Postfachs zum Löschen vor.

int **imap_delete** (int imap_stream, int msg_number [, int flags])

Diese Funktion liefert immer `TRUE`, eventuelle Fehler können nicht erkannt werden!

Die **imap_delete()** Funktion merkt die mit *`msg_number`* angegebene Nachricht des aktuellen Postfachs zum Löschen vor. Enthält der optionale Parameter *`flags`* den Wert *`FT_UID`*, so wird der Parameter *`msg_number`* als *`UID`* interpretiert. Die Nachricht verbleibt zunächst noch im Postfach, endgültig gelöscht wird sie (zusammen mit allen anderen entsprechend markierten Nachrichten) durch Aufruf der Funktion imap_expunge() oder durch imap_close() mit dem optionalen Flag *`CL_EXPUNGE`*.

Beispiel 1. imap_delete() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    || die ("can't connect: " . imap_last_error());

$check = imap_mailboxmsginfo ($mbox);
print "Messages before delete: " . $check->Nmsgs . "<br>\n" ;
imap_delete ($mbox, 1);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after  delete: " . $check->Nmsgs . "<br>\n" ;
imap_expunge ($mbox);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after expunge: " . $check->Nmsgs . "<br>\n" ;
```

```
imap_close ($mbox);
```

imap_deletemailbox (PHP 3, PHP 4 >= 4.0.0)

Löscht ein Postfach

int **imap_deletemailbox** (int imap_stream, string mbox)

imap_deletemailbox() löscht das angegebene Postfach inclusive aller darin enthaltener Nachrichten und Unter-Postfächer.

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

Siehe auch imap_createmailbox() und imap_renamemailbox() sowie imap_open() für das Format des *mbox* Parameters.

imap_expunge (PHP 3, PHP 4 >= 4.0.0)

Löscht alle zum Löschen vorgemerkten Nachrichten

int **imap_expunge** (int imap_stream)

imap_expunge() löscht endgültig alle von den Funktionen imap_delete(), imap_mail_move() oder imap_setflag_full() zum Löschen vorgemerkten Nachrichten.

Diese Funktion liefert immer TRUE, eventuelle Fehler können nicht erkannt werden!

imap_fetchbody (PHP 3, PHP 4 >= 4.0.0)

Liefert einen bestimmten Abschnitt aus dem Körper einer Nachricht.

string **imap_fetchbody** (int imap_stream, int msg_number, string part_number [, flags flags])

Diese Funktion liefert einen bestimmten Teil einer MIME-kodierten Nachricht als String. Die Nummer des gewünschten Teils besteht aus von Punkten getrennten Zahlen, die den Index des Teils gemäß der IMAP4-Spezifikationen bezeichnen. Der Inhalt wird dabei von dieser Funktion nicht dekodiert.

Die Bitmaske *flags* kann aus folgenden Werten zusammengesetzt werden:

- FT_UID - *msg_number* enthält eine UID
- FT_PEEK - Das \Seen Flag der Nachricht wird nicht verändert
- FT_INTERNAL - Das Ergebnis wird ohne jeden Versuch einer Konvertierung der Zeilenenden zurückgegeben

imap_fetchstructure (PHP 3, PHP 4 >= 4.0.0)

Liefert die Struktur der angegebenen Nachricht

object **imap_fetchstructure** (int imap_stream, int msg_number [, int flags])

Diese Funktion liefert die gesamten Strukturinformationen zu einer gegebenen Nachricht. Das zurückgegebene Objekt enthält den Umschlag, die internen Daten, Größe und Flags der Nachricht sowie ein (rekursives) Array von ähnlichen Objekten für jeden enthaltenen MIME-Teil der Nachricht. Enthält der optionale Parameter `flags` den Wert `FT_UID`, so wird `msg_number` als UID anstelle eines Index interpretiert.

Das zurückgegebene Objekt verfügt über die folgenden Attribute:

Tabelle 1. Zurückgegebene Objekte von imap_fetchstructure()

type	Primärer MIME-Typ des Nachrichten-Teils
encoding	Transfer-Kodierung
ifsubtype	Wahr, wenn das Feld subtype gesetzt ist
subtype	MIME Subtyp
ifdescription	Wahr, wenn ein Beschreibungsstring existiert
description	Inhalt des Bechreibungsstrings
ifid	Wahr, wenn ein Identifikationsstring existiert
id	Identificationsstring
lines	Anzahl der Zeilen dieses Teils
bytes	Größe des Teils in Byte
ifdisposition	Wahr, wenn der Dispositionsstring gesetzt ist
disposition	Dispositionsstring
ifdparameters	Wahr, wenn Dispositions-Parameterwerte gesetzt sind
dparameters	Array der DispositionsParameter
ifparameters	Wahr, wenn Parameter gesetzt sind
parameters	Array der MIME Parameter
parts	Array von Objekten, die die einzelnen Unterteile der Nachricht beschreiben

1. dparameters ist ein Array von Objekten mit jeweils einem "attribute" und einem "value" Attribut.
2. parameters ist ein Array von Objekten mit jeweils einem "attribute" und einem "value" Attribut.
3. Die Elemente des parts Array entsprechen dem von der Funktion zurückgegebenen Objekt (usf.).

Tabelle 2. Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Tabelle 3. Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

imap_header (PHP 3, PHP 4 >= 4.0.0)

Liefert den Kopf einer Nachricht

object **imap_header** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]])

Diese Funktion liefert ein Objekt, das aus verschiedenen Header-Elementen gebildet wird.

```
    remail, date, Date, subject, Subject, in_reply_to, message_id,
    newsgroups, followup_to, references

message flags:
  Recent -  'R' wenn neu und bereits gelesen,
            'N' wenn neu und noch nicht gelesen,
            ' ' wenn nicht neu
  Unseen -  'U' wenn noch nicht gelesen UND nicht neu,
            ' ' wenn gelesen OR neu und noch nicht gelesen
  Answered -'A' wenn beantwortet,
            ' ' sonst
  Deleted - 'D' wenn zum Löschen vorgemerkt
            ' ' sonst
  Draft -   'X' wenn als Entwurf markiert,
```

```
        ' ' sonst
  Flagged - 'F' wenn als 'wichtig' markiert,
        ' ' sonst
```

Beachten Sie bitte die etwas seltsame Recent/Unseen-Zuordnung.
Eine Nachricht wurde noch nicht gelesen, wenn folgende Bedingung
erfüllt ist:

Unseen == 'U' || Recent == 'N'

toaddress (Inhalt der to: Zeile, bis zu 1024 Zeichen)

to[] (ein Array aus Objekten aus der to: Zeile, jeweils mit:)
  personal
  adl
  mailbox
  host

fromaddress (Inhalt der from: Zeile, up to 1024 characters)

from[] (Inhalt der From Zeile, bestehend aus:)
  personal
  adl
  mailbox
  host

ccaddress (Inhalt der cc: Zeile, max. 1024 Zeichen)
cc[] (Inhalt der Cc Zeile, bestehend aus:)
  personal
  adl
  mailbox
  host

bccaddress (Inhalt der bcc Zeile, max. 1024 Zeichen)
bcc[] (Inhalt der Bcc Zeile, bestehend aus:)
  personal
  adl
  mailbox
  host

reply_toaddress (Inhalt der reply_to: Zeile, max. 1024 Zeichen)
reply_to[] (Inhalt der Reply_to Zeile, bestehend aus:)
  personal
  adl
  mailbox
  host

senderaddress (Inhalt der sender: Zeile, max. 1024 Zeichen)
sender[] (Inhalt der sender Zeile, bestehend aus:)
  personal
  adl
  mailbox

host

return_path (Inhalt der return-path: Zeile, max. 1024 Zeichen)
return_path[] (Inhalt der return_path Zeile, bestehend aus:)
personal
adl
mailbox
host

udate (Sende-Zeitpunkt der Nachricht in Sekunden seit 1.1.1970)

fetchfrom (From: Zeile, formartiert auf
max. `fromlength` Zeichen)

fetchsubject (Subject: Zeile, formatiert auf
max. `subjectlength` Zeichen)

imap_headers (PHP 3, PHP 4 >= 4.0.0)

Liefert eine Zusammenfassung aller Nachrichtenköpfe eines Postfachs

array **imap_headers** (int imap_stream)

Diese Funktion liefert ein String-Array, das je Nachricht einen String mit vorformatierten Informationen aus dem Nachrichtenheader enthält.

imap_listmailbox (PHP 3, PHP 4 >= 4.0.0)

Liefert eine Liste der Postfach-Namen

array **imap_listmailbox** (int imap_stream, string ref, string pat)

Liefert die Namen der gefundenen Postfächer als String-Array. `ref` und `pat` werden in imap_getmailboxes() beschrieben.

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Liefert detailierte Informationen über eine Auswahl von Postfächern

array **imap_getmailboxes** (int imap_stream, string ref, string pat)

imap_getmailboxes() liefert ein Array zurück, das für jedes gefundene Postfach ein Objekt mit de Attributen `name`, `delimiter` und `attributes` enthält. `name` enthält dabei den vollständigen Pfadnamen des Postfachs, `delimiter` das verwendete Trennzeichen zwischen Postfach-Ebenen,

und *attributes* besteht aus einer Bitmaske, die einzelne Eigenschaften des Postfachs beschreibt:

- LATT_NOINFERIORS - es existieren keine weiteren Postfächer unterhalb dieses Postfachs
- LATT_NOSELECT - Dieses "Postfach" ist nur ein Behälter für weitere Postfächer.

  Es kann selbst keine Nachrichten enthalten und somit auch nicht geöffnet werden.

- LATT_MARKED - Dieses Postfach wurde markiert (nur bei UW-IMAPD).
- LATT_UNMARKED - Dieses Postfach wurde nicht markiert (nur bei UW-IMAPD).

Namen von Postfächern, die Zeichen außerhalb des druckbaren ASCII-Bereiches enthalten, können mit Hilfe der Funktion imap_utf7_decode() dekodiert werden.

ref sollte normalerweise nur die Spezifikation des Servers in der in imap_open() beschriebenen Form enthalten. Über *pat* kann ein Startpunkt innerhalb der Postfach-Hierachie vorgegeben werden. Wünschen Sie die Ausgabe aller Postfächer, so übergeben Sie als *pat* *'*'*.

Die Zeichen '*' und '%' haben innerhalb von *pattern* eine besondere Bedeutung:

- '*' - entspricht beliebig vielen Zeichen (incl. des Trennzeichens zw. Postfach-Namen)
- '%' - entspricht beliebig vielen Zeichen innerhalb eines Postfach-Namens

So liefert "*" alle verfügbaren Postfächer auf dem Server, "INBOX/*" dementsprechend alle persönlichen Postfächer des aktuellen Benutzers. "%" dagegen liefert nur die Toplevel-Postfächer des Servers, "INBOX/%" dementsprechend nur die Toplevel-Postfächer des aktuellen Benutzers. Ein '%' am Ende von *pattern* entspricht zusätzlich auch allen Zeichenfolgen, die mit dem Trennzeichen beginnen.

Beispiel 1. imap_getmailboxes() Beispiel

```
$mbox = imap_open ("{your.imap.host}", "username", "password", OP_HALFOPEN)
    || die ("can't connect: " . imap_last_error());

$list = imap_getmailboxes ($mbox, "{your.imap.host}", "*");
if (is_array ($list)) {
    reset ($list);
    while (list( $key, $val) = each ($list)) {
        print "($key) ";
        print imap_utf7_decode ($val->name) . ",";
        print "'" . $val->delimiter . "',";
        print $val->attributes . "<br>\n";
    }
} else {
    print "imap_getmailboxes failed: " . imap_last_error() . "\n";
}
imap_close ($mbox);
```

imap_listsubscribed (PHP 3, PHP 4 >= 4.0.0)

Liefert eine Liste aller abbonierten Postfächer

array **imap_listsubscribed** (int imap_stream, string ref, string pattern)

Diese Funktion entspricht weitgehend imap_listmailbox(), allerdings werden nur Postfächer zurückgegeben, die der aktuelle Benutzer auch abboniert hat.

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Liefert eine Auswahl aller abbonierten Postfächer

array **imap_getsubscribed** (int imap_stream, string ref, string pattern)

Diese Funktion entspricht weitgehend imap_getmailboxes(), allerdings werden nur Postfächer zurückgegeben, die der aktuelle Benutzer auch abboniert hat.

imap_mail_copy (PHP 3, PHP 4 >= 4.0.0)

Kopiert Nachrichten in ein Postfach

int **imap_mail_copy** (int imap_stream, string msglist, string mbox [, int flags])

Diese Funktion kopiert die in *msglist* angegebenen Nachrichten in das Postfach *mbox*. *msglist* kann eine einzelne Nachrichtennummer oder Aufzählungen und Bereiche gemäß RFC2060 (http://www.faqs.org/rfcs/rfc2060.html) enthalten.

Die Bitmaske *flags* kann aus den foldengen Werten zusammengesetzt werden.

- CP_UID - Die Nummern in *msglist* entsprechen den UIDs der gewünschten Nachrichten
- CP_MOVE - Die ursprunglichen Nachrichten werden nach dem Kopieren gelöscht. copying

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

imap_mail_move (PHP 3, PHP 4 >= 4.0.0)

Verschiebt Nachrichten in ein anderes Postfach

int **imap_mail_move** (int imap_stream, string msglist, string mbox [, int flags])

Diese Funktion verschiebt die in *msglist* angegebenen Nachrichten in das Postfach *mbox*. *msglist* kann eine einzelne Nachrichtennummer oder Aufzählungen und Bereiche gemäß RFC2060 (http://www.faqs.org/rfcs/rfc2060.html) enthalten.

Die Bitmaske *flags* kann den foldengen Wert enthalten:

- CP_UID - Die Nummern in *msglist* entsprechen den UIDs der gewünschten Nachrichten

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

imap_num_msg (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Nachrichten im aktuellen Postfach

int **imap_num_msg** (int imap_stream)

Liefert die Anzahl der Nachrichten im aktuellen Postfach

imap_num_recent (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der neu hinzugekommenen Nachrichten im aktuellen Postfach

int **imap_num_recent** (int imap_stream)

Liefert die Anzahl der neu hinzugekommenen Nachrichten im aktuellen Postfach

imap_open (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine Verbindung zu einem Postfach auf einem POP-, IMAP- oder NNTP-Server

int **imap_open** (string mailbox, string username, string password [, int flags])

Diese Funktion liefert ein Verbindungs-Handle bzw. FALSE bei evtl. auftretenden Fehlern. Verbindungen können nicht nur zu IMAP-, sondern auch zu POP- oder NNTP-Servern aufgebaut werden, wobei dann allerdings nicht der gesamte Funktionsumfang zur Verfügung steht. Liefert ein IMAP-Streamhandle oder FALSE bei Fehlern.

Der Name eines Postfachs setzt sich zusammen aus der Bezeichnung des Servers und dem Pfad des Postfachs auf diesem Server. Der spezielle Name INBOX steht dabei als Synonym für das persönliche Postfach des jeweiligen Benutzers. Die Bezeichnung des Servers setzt sich zusammen aus dem eigentlichen Servernamen oder der entsprechenden IP-Adresse sowie das gewünschte Protokoll (getrennt durch '/') und optional der zu benutzende Port (getrennt durch ':'). Die gesamte Serverbezeichnung wird in '{' und '}' eingeschlossen und muß immer mit angegeben werden. Enthält der Name eines Postfachs Zeichen außerhalb des druckbaren ASCII-Bereiches wie z.B. Umlaute oder das 'ß', so muß er mit Hilfe der Funktion imap_utf7_encode() codiert werden.

Der optionale Parameter *flags* kann sich aus den folgenden Bitmasken zusammensetzen:

- OP_READONLY - Über diese Verbindung darf nur lesend zugegriffen werden
- OP_ANONYMOUS - Bei NNTP-Verbindungen wird die Datei .newsrc nicht aktualisiert
- OP_HALFOPEN - es wird eine IMAP- oder NNTP-Verbindung aufgebaut, aber noch kein Postfach ausgewählt

- CL_EXPUNGE - beim Schließen der Verbindung werden alle zum Löschen vorgemerkte Nachrichten endgültig gelöscht (siehe auch imap_delete() und imap_expunge())

Das folgende Beispiel stellt eine Verbindung zu einem lokalen IMAP-Server (IMAP = Port 143) her:

```
$mbox = imap_open ("{localhost:143}INBOX", "user_id", "password");
```

Eine Verbindung zu einem lokalen POP3-Server (POP3 = Port 110) erhalten sie mit:

```
$mbox = imap_open ("{localhost/pop3:110}INBOX", "user_id", "password");
```

Eine Verbindung zu einem lokalen News-Server (NNTP = Port 119) erhalten Sie mit:

```
$nntp = imap_open ("{localhost/nntp:119}comp.test", "", "");
```

Eine Verbindung zu einem nicht lokalen Server können Sie herstellen, indem Sie in den oben genannten Beispielen "localhost" durch den Namen oder die IP-Adresse des Servers ersetzen.

Beispiel 1. imap_open() Beispiel

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password");

echo "<p><h1>Mailboxes</h1>\n";
$folders = imap_listmailbox ($mbox, "{your.imap.host:143}", "*");
if ($folders == false)
    echo "Call failed<br>\n";
else
    while (list ($key, $val) = each ($folders))
        echo $val . "<br>\n";

echo "<p><h1>Headers in INBOX</h1>\n";
$headers = imap_headers ($mbox);
if ($headers == false)
    echo "Call failed<br>\n";
else
    while (list ($key, $val) = each ($headers))
        echo $val . "<br>n";

imap_close ($mbox);
```

imap_ping (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob die aktuelle Verbindung noch verfügbar ist

int **imap_ping** (int imap_stream)

Liefert TRUE, wenn die aktuelle Verbindung noch verfügbar ist, sonst FALSE.

Die **imap_ping()** Funktion schickt ein Ping-Signal an den Mailserver und prüft so die Verfügbarkeit der Verbindung. Auf diese Weise kann die Verbindung zu einem Server aufrechterhalten werden, der Verbindungen nach einer gewissen Zeit ohne Aktivität automatisch beendet (da PHP-Scripte i.A. nicht sehr lange laufen, kann ich mir kaum vorstellen, das jemand diese Funktion wirklich benötigt).

imap_renamemailbox (PHP 3, PHP 4 >= 4.0.0)

Ändert den Namen eines Postfachs

int **imap_renamemailbox** (int imap_stream, string old_mbox, string new_mbox)

Die Funktion **imap_renamemailbox()** ändert den Namen eines Postfachs.

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

Siehe auch imap_createmailbox() und imap_deletemailbox() sowie imap_open() für das Format der Parameter *old_mbox* und *new_mbox*.

imap_reopen (PHP 3, PHP 4 >= 4.0.0)

Wechselt das aktuelle Postfach der Verbindung

int **imap_reopen** (string imap_stream, string mailbox [, string flags])

Mit Hilfe dieser Funktion kann auf IMAP- und NNTP-Servern ein anderes Postfach als das bisher aktuelle ausgewählt werden.

Der optionale Parameter *flags* kann sich aus den folgenden Bitmasken zusammensetzen:

- OP_READONLY - Auf das Postfach soll nur lesend zugegriffen werden
- OP_ANONYMOUS - Bei NNTP-Verbindungen wird die persönliche .newsrc Datei nicht aktualisiert
- OP_HALFOPEN - bei IMAP- oder NNTP-Servern wird in den allgemeinen Verbindungsstatus ohne aktuelles Postfach zurückgeschaltet
- CL_EXPUNGE - beim Schließen der Verbindung werden alle zum Löschen vorgemerkte Nachrichten endgültig gelöscht (siehe auch imap_delete() und imap_expunge())

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

imap_subscribe (PHP 3, PHP 4 >= 4.0.0)

Abbonieren eines Postfachs

int **imap_subscribe** (int imap_stream, string mbox)

Abbonieren eines Postfachs.

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

imap_undelete (PHP 3, PHP 4 >= 4.0.0)

Nimmt eine bereits gesetzte Löschmarkierung einer Nachricht zurück

int **imap_undelete** (int imap_stream, int msg_number)

Diese Funktion nimmt eine evtl. durch die Funktionen imap_delete() oder imap_mail_move() gesetzte Löschmarkierung für die durch *msg_number* spezifizierte Nachricht zurück.

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

imap_unsubscribe (PHP 3, PHP 4 >= 4.0.0)

Abbonement eines Postfachs beenden

int **imap_unsubscribe** (int imap_stream, string mbox)

Abbonement eines Postfachs beenden.

Liefert TRUE bei Erfolg und FALSE bei Fehlern.

imap_qprint (PHP 3, PHP 4 >= 4.0.0)

Konvertiert einen quoted-printable kodierten String in einen 8bit-String

string **imap_qprint** (string string)

Konvertiert einen quoted-printable kodierten String (gemäß RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Abschnitt 6.7) in einen 8bit-String.

Siehe auch imap_8bit().

imap_8bit (PHP 3, PHP 4 >= 4.0.0)

Konvertiert einen String in einen quoted-printable kodierten String.

string **imap_8bit** (string string)

Konvertiert einen String in einen quoted-printable kodierten String (gemäß RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Abschnitt 6.7).

Siehe auch imap_qprint().

imap_binary (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Konvertiert 8Bit-Text in einen BASE64-kodierten String.

string **imap_binary** (string string)

Konvertiert 8Bit-Text in einen BASE64-kodierten String (gemäß RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Abschnitt 6.8).

Siehe auch imap_base64().

imap_scanmailbox (PHP 3, PHP 4 >= 4.0.0)

Druchsucht Postfächer nach einen String

array **imap_scanmailbox** (int imap_stream, string ref, string pat, string content)

Diese Funktion arbeitet ähnlich wie imap_listmailbox(). Sie gibt aber nur die Namen der Postfächer zurück, in denen irgendwo der Text `content` vorkommt.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Liefert Informationen über das aktuelle Postfach

object **imap_mailboxmsginfo** (int imap_stream)

Diese Funktion liefert dieselben Informationen wie imap_status() sowie zusätzlich die Größe des Postfachs in Bytes. Diese wird aus der Summe der Größe der einzelnen Nachrichten gebildet, deshalb benötigt diese Funktion mehr Zeit zur Ausführung als imap_status().

Tabelle 1. Felder des Rückgabe-Objekts

Date	Datum der letzten Änderung
Driver	Driver
Mailbox	Name des Postfachs
Nmsgs	Anzahl der Nachrichten
Recent	Anzahl neuer Nachrichten
Unread	Anzahl ungelesener Nachrichten
Size	Größe des Postfachs in Byte

Beispiel 1. imap_mailboxmsginfo() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    || die ("can’t connect: " . imap_last_error());

$check = imap_mailboxmsginfo ($mbox);
```

```
if ($check) {
    print "Date: "     . $check->Date    . "<br>\n";
    print "Driver: "   . $check->Driver  . "<br>\n";
    print "Mailbox: "  . $check->Mailbox . "<br>\n";
    print "Messages: " . $check->Nmsgs   . "<br>\n" ;
    print "Recent: "   . $check->Recent  . "<br>\n";
    print "Size: "     . $check->Size    . "<br>\n";
} else {
    print "imap_check() failed: " . imap_lasterror() . "<br>\n";
}
imap_close ($mbox);
```

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Bildet aus Realnamen, Postfach und Server eine korekt formatierte Mail-Adresse

string **imap_rfc822_write_address** (string mailbox, string host, string personal)

Bildet aus Realnamen, Postfach und Server eine korekt formatierte Mail-Adresse gemäß RFC822 (http://www.faqs.org/rfcs/rfc822.html).

Beispiel 1. imap_rfc822_write_address() Beispiel

```
print imap_rfc822_write_address ("hartmut", "cvs.php.net",
                                 "Hartmut Holzgraefe);
```

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Parsen eines Adress-Strings

array **imap_rfc822_parse_adrlist** (string address, string default_host)

Diese Funktion zerlegt einen Adress-String gemäß RFC822 (http://www.faqs.org/rfcs/rfc822.html) und liefert ein Array von Objekten mit einem Eintrag je erkannter Adresse. Die Address-Objekte bestehen dabei jeweils aus den folgenden vier String-Elementen:

- mailbox - Name des Postfachs (Username)
- host - Hostname
- personal - Realer Name des Empfängers
- adl - at domain source route

Beispiel 1. imap_rfc822_parse_adrlist() Beispiel

```
$address_string = "Hartmut Holzgraefe <hartmut@php.net>,
                   postmaster@somedomain.net, root";
$address_array  = imap_rfc822_parse_adrlist ($address_string,
                                             "somedomain.net");
if (! is_array ($address_array)) die ("somethings wrong\n");

reset ($address_array);
while (list( $key, $val) = each ($address_array)) {
    print "mailbox : " . $val->mailbox  . "<br>\n";
    print "host    : " . $val->host     . "<br>\n";
    print "personal: " . $val->personal . "<br>\n";
    print "adl     : " . $val->adl      . "<p>\n";
}
```

imap_setflag_full (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Setzt Flags einer Nachricht

string **imap_setflag_full** (int stream, string sequence, string flag, string options)

Diese Funktion setzt die übergebenen Flags für eine oder mehrere Nachrichten.

Mögliche Flags sind "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", und "\\Recent" (siehe auch RFC2060 (http://www.faqs.org/rfcs/rfc2060.html))

Enthält *options* den Wert ST_UID, so werden die übergebenen Nachrichtennummern als UIDs anstelle von Indizes interpretiert.

Beispiel 1. imap_setflag_full() Beispiel

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password")
     || die ("can't connect: " . imap_last_error());

$status = imap_setflag_full ($mbox, "2,5", "\\Seen \\Flagged");
if (!$status) echo imap_last_error();

imap_close ($mbox);
```

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Löscht Flags einer Nachricht

string **imap_clearflag_full** (int stream, string sequence, string flag, string options)

Diese Funktion löscht die übergebenen Flags für eine oder mehrere Nachrichten.

Mögliche Flags sind "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", und "\\Recent" (siehe auch RFC2060 (http://www.faqs.org/rfcs/rfc2060.html))

Enthält `options` den Wert ST_UID, so werden die übergebenen Nachrichtennummern als UIDs anstelle von Indizes interpretiert.

imap_sort (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Sortiert Nachrichten eines Postfachs

array **imap_sort** (int stream, int criteria, int reverse, int options)

Diese Funktion liefert die Indizes der Nachrichten eines Postfachs als sortiertes Array zurück. Als Sortier-Kriterium kann im Parameter `criteria` genau einer der folgenden Werte angegeben werden

- SORTDATE Versanddatum der Nachricht
- SORTARRIVAL Empfangsdatum der Nachricht
- SORTFROM Benutzername des ersten From: Eintrags
- SORTSUBJECT Betreff der Nachricht
- SORTTO Benutzername des ersten To: Eintrags
- SORTCC Benutzername des ersten CC: Eintrags
- SORTSIZE Größe der Nachricht in Bytes

Ist `reverse` ungleich 0, so werden die Nachrichten in umgekehrter Reihenfolge übergeben.

Die Bitmaske `options` kann aus folgenden werten zusammengesetzt werden:

- SE_UID Es werden UIDs anstelle von Indizes zurückgegeben
- SE_NOPREFETCH Es werden keine Nachrichten im Cache gehalten

imap_fetchheader (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Liefert den Kopf einer Nachricht

string **imap_fetchheader** (int imap_stream, int msgno, int flags)

Diese Funktion liefert den vollständigen, ungefilterten und gemäß RFC 822 (http://www.faqs.org/rfcs/rfc822.html) formatierten Kopf einer Nachricht als String zurück.

Die Bitmaske `flags` kann aus folgenden Werten zusammengesetzt werden:

- FT_UID `msgno` enthält eine UID

- FT_INTERNAL Der Kopf wird im "internen" Format, d.h. ohne jegliche Konvertierung von Zeilenenden zurückgegeben
- FT_PREFETCHTEXT Der Körper der Nachricht wird ebenfalls angefordert, hierdurch wird ein zus. Zugriff vermieden, wenn z.B. für eine Save-Operation der gesamte Text einer Nachricht benötigt wird

imap_uid (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Liefert die UID zu einem gegeben Nachrichten-Index

int **imap_uid** (int imap_stream, int msgno)

Diese Funktion ermittelt die UID zu einem gegebenen Nachrichten-Index. Während die Nachrichten-Indices einfach von eins an aufsteigend durchnummeriert sind und sich z.B. bei Löschoperationen ändern können, ist die UID einer Nachricht innerhalb eines Postfachs unveränderbar eindeutig. (Siehe auch imap_msgno()).

imap_msgno (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Liefert den aktuellen Nachrichten-Index zu einer UID

int **imap_msgno** (int imap_stream, int uid)

Diese Funktion liefert den aktuellen Index einer Nachricht innerhalb eines Postfachs aus der eindeutigen UID. Sie ist somit die Umkehrung der Funktion imap_uid().

imap_search (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Sucht Nachrichten, die den übergebenen Suchkriterien entsprechen

array **imap_search** (int imap_stream, string criteria, int flags)

Diese Funktion durchsucht die Nachrichten des aktuellen Postfachs und liefert ein Array mit den Nummern der Nachrichten, die mit den angegebenen Kriterien übereinstimmen. Der Parameter `criteria` enthält einen String, der sich aus den folgenden Suchbegriffen zusammensetzen kann

- ALL - liefert alle Nachrichten, die den restlichen Kriterien entsprechen
- ANSWERED - das \\ANSWERED Flag der Nachricht ist gesetzt
- BCC "string" - Das Bcc: Feld der Nachricht enthält "string"
- BEFORE "date" - Das Date: Feld der Nachricht enthält ein Datum, das vor dem in "date" angegebenen liegt
- BODY "string" - Im Inhalt der Nachricht ist "string" enthalten
- CC "string" - Das Cc: Feld der Nachricht enthält "string"
- DELETED - Die Nachricht ist zum Löschen vorgemerkt

- FLAGGED - Das \\FLAGGED (wichtig bzw. dringend) Flag der Nachricht ist gesetzt
- FROM "string" - Das From: Feld der Nachricht enthält "string"
- KEYWORD "string" - Das Feld Keyword: enthält "string"
- NEW - die Nachricht ist neu
- OLD - die Nachricht ist nicht neu
- ON "date" - Das Datum in Date: entspricht "date"
- RECENT - Das \\RECENT Flag ist gesetzt
- SEEN - Das \\SEEN Flag ist gesetzt, die Nachricht wurde bereits einmal abgerufen
- SINCE "date" - Das Datum "date" liegt vor dem im Date: Feld der Nachricht
- SUBJECT "string" - "string" ist Teil des Betreffs der Nachricht
- TEXT "string" - "string" ist Teil der Nachricht (Kopf oder Inhalt)
- TO "string" - Das To: Feld der Nachricht enthält "string"
- UNANSWERED - Die Nachricht wurde nicht beantwortet, das \\Answered Flag ist nicht gesetzt
- UNDELETED - Die Nachricht ist nicht zum Löschen vorgemerkt
- UNFLAGGED - Die Nachricht ist nicht als 'wichtig' markiert
- UNKEYWORD "string" - "string" ist nicht Schlüsselwort der Nachricht
- UNSEEN - Die Nachricht wurde bisher nicht gelesen

Argumente, die Leerzeichen enthalten, müssen dabei in Anführungszeichen eingebettet werden.

Um zum Beispiel alle unbeantworteten Nachrichten von Mama zu finden übergeben Sie als `criteria`: "UNANSWERED FROM Mama". Die Suche scheint nicht zwischen Groß- und Kleinschreibung zu unterscheiden. Die Liste der Suchbegriffe entstand nach Inspektion der UW c-client Sourcen und ist vermutlich nicht vollständig (siehe auch RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), Abschnitt 6.4.4).

Enthält der optionale Parameter `flags` den Wert SE_UID, so werden UIDs anstelle von laufenden Nachrichtennummern zurückgegeben.

imap_last_error (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Diese Funktion liefert die zuletzt während dieses Seitenzugriffs aufgetretene Fehlermeldung (falls vorhanden)

string **imap_last_error** (void)

Diese Funktion liefert den vollständigen Text der zuletzt aufgetretenen Fehlermeldung dieses Requests. Der Zustand des Fehlerstacks wird durch diese Funktion nicht verändert, weitere Aufrufe von **imap_last_error()** liefern somit weiter dasselbe Ergebis, solange keine weiteren Fehler aufgetreten sind.

imap_errors (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Diese Funktion liefert alle bisher aufgetretenen Fehlermeldungen

array **imap_errors** (void)

Diese Funktion liefert in Form eines String-Arrays alle seit Beginn des Requests bzw. seit dem letzten Aufruf von **imap_errors()** aufgetretenen Fehlermeldung. Der Fehler-Stack wird dabei gelöscht, so daß folgende Aufrufe von **imap_errors()** bzw. **imap_last_errors()** bis zum mächsten aufgetretenen Fehler kein Ergebnis liefern.

imap_alerts (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Diese Funktion liefert alle alert-Meldungen zurück, die während der Ausführung dieses Requests bzw. seit dem letzten Reset des alert-Stacks aufgetreten sind.

array **imap_alerts** (void)

Diese Funktion liefert in Form eines String-Arrays alle IMAP alert-Medlungen, die seit Beginn dieses Requests bzw. dem letzten Aufruf der Funktion aufgetreten sind. Durch den Aufruf der Funktion wird der alert-Stack gelöscht, so daß weitere Aufrufe zunächst bis zum Auftreten eines neuen alerts kein Ergebnis liefern.

Die IMAP-Spezifikationen verlangen, daß diese Meldungen an den Benutzer weitergegeben werden.

imap_status (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Liefert ausgewählte Statusinformationen zum angegebenen Postfach

object **imap_status** (int imap_stream, string mailbox, int options)

Diese Funktion liefert Statusinformationen zum mit `mailbox` angegebenen Postfach. Im der Bitmaske `options` können dabei folgende Funktionen ausgewählt werden, die den Inhalt des zurückgegebenen Objekts beeinflussen:

- SA_MESSAGES - das Feld messages wird mit der Anzahl der enthaltenen Nachrichten gefüllt
- SA_RECENT - das Feld recent wird mit der Anzahl der neuen Nachrichten gefüllt
- SA_UNSEEN - das Feld unseen wird mit der Anzahl der noch nicht abgerufenen Nachrichten gefüllt
- SA_UIDNEXT - das Feld uidnext wird mit der UID gefüllt, die die nächste in diesem Postfach abgelegte Nachricht erhalten wird
- SA_UIDVALIDITY - das Feld uidvalidity wird mit einer Konstanten gefüllt, die sich immer dann verändern wird, wenn die UIDs der Nachrichten im Postfach sich vermutlich verändert haben
- SA_ALL - es werden alle beschriebenen Felder gefüllt

Status->flags wird ebenfalls gesetzt und enthält eine Bitmaske aus den oben genannten Werten, die die tatsächlich zurückgegebenen Informationen beschreibt.

Beispiel 1. imap_status() Beispiel

```
$mbox = imap_open ("{your.imap.host}", "username", "password", OP_HALFOPEN)
      || die ("can't connect: " . imap_last_error());

$status = imap_status ($mbox, "{your.imap.host}INBOX", SA_ALL);
if($status) {
    print ("Messages:    " . $status->messages   ) . "<br>\n";
    print ("Recent:      " . $status->recent     ) . "<br>\n";
    print ("Unseen:      " . $status->unseen     ) . "<br>\n";
    print ("UIDnext:     " . $status->uidnext    ) . "<br>\n";
    print ("UIDvalidity: " . $status->uidvalidity) . "<br>\n";
} else {
    print "imap_status failed: " . imap_lasterror() . "\n";
}
imap_close ($mbox);
```

imap_utf7_decode (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Dekodiert einem String im modifizierten UTF-7 Format.

string **imap_utf7_decode** (string text)

Dekodiert den String *text* aus dem modifizierten UTF-7 Format in 8Bit-Text.

Liefert den dekodierten Text oder FALSE, wenn die Eingabe nicht gemäß modifiziertem UTF-7 formatiert war. Diese Funktion wird benötigt, um Postfachnamen zu decodieren, die internationale Sonderzeichen, wie z.B. unsere Umlaute, enthalten.

(siehe RFC2152 (http://www.faqs.org/rfcs/rfc2152.html) für UTF-7 und RFC2060 (http://www.faqs.org/rfcs/rfc2060.html), Abschnitt 5.1.3 für modifiziertes UTF-7)

imap_utf7_encode (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Kodiert Text im modifizieren UTF-7 Format

string **imap_utf7_encode** (string data)

Konvertiert den in *data* übergebenen Text in das modifizierte UTF-7 Format. Diese Funktion wird benötigt, um Postfachnamen darzustellen, die internationale Sonderzeichen, wie z.B. unsere Umlaute, enthalten.

(siehe RFC2152 (http://www.faqs.org/rfcs/rfc2152.html) für UTF-7 und RFC2060 (http://www.faqs.org/rfcs/rfc2060.html), Abschnitt 5.1.3 für modifiziertes UTF-7)

imap_utf8 (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Konvertiert Text zu UTF8

string **imap_utf8** (string text)

Konvertiert den übergebenen String in das UTF8-Format gemäß RFC2044 (http://www.faqs.org/rfcs/rfc2044.html).

imap_fetch_overview (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Liefert einen Auszug aus den Header-Feldern von Nachrichten

array **imap_fetch_overview** (int imap_stream, string sequence [, int flags])

Diese Funktion liefert ausgewählte Headerfelder der in `sequence` übergebenen Nachrichten und liefert sie in einem Array mit je einem Objekt pro Nachricht zurück. `sequence` enthält eine Sequenz von Nachrichten-Indizes oder, bei gesetztem FT_UID in `flags`, von UIDs. Die einzelnen Objekte des zurückgegeben Arrays enthalten dabei folgende Informationen:

- subject - Betreff
- from - Absender
- date - Sendedatum
- message_id - Message-ID
- references - bezieht sich auf Message-ID
- size - Größe in Byte
- uid - UID der Nachricht im Postfach
- msgno - Index der Nachricht im Postfach
- recent - Flag gesetzt
- flagged - Flag gesetzt
- answered - Flag gesetzt
- deleted - Flag gesetzt
- seen - Flag gesetzt
- draft - Flag gesetzt

Beispiel 1. imap_fetch_overview() Beispiel

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password")
    || die ("can’t connect: " . imap_last_error());

$overview = imap_fetch_overview ($mbox, "2,4:6", 0);

if (is_array ($overview )) {
    reset ($overview);
    while (list ($key, $val) = each ($overview)) {
```

```
            print $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
    }

    imap_close ($mbox);
```

imap_mime_header_decode (PHP 3>= 3.0.17, PHP 4 >= 4.0.0)

Decode MIME header elements

array **imap_header_decode** (string text)

imap_mime_header_decode() decodiert MIME Headerfelder, die mehr als nur reinen ASCII-Text enthalten und gemäß RFC2047 (http://www.faqs.org/rfcs/rfc2047.html) codiert wurden. Die decodierten Teilstücke werden in einem Array von Objekten zurückgegeben, wobei jedes Objekt die Felder "text" für den decodierten Teiltext und "charset" für den zugehörigen Zeichensatz enthält. Nichtcodierte Teile des Headerfeldes werden dabei durch den Wert "default" beschrieben.

Im folgenden Beispiel besteht das Headerfeld aus zwei Teilstücken, wobei das erste nach ISO-8859-1 codiert ist:

Beispiel 1. imap_mime_header_decode() example

```
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";

$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
       echo "Charset: {$elements[$i]->charset}\n";
       echo "Text: {$elements[$i]->text}\n\n";
}
```

imap_mail_compose (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Erzeugt eine MIME-Nachricht aus gegebenen Header- und Body-Teilen

string **imap_mail_compose** (array envelope, array body)

imap_mail (PHP 3>= 3.0.14, PHP 4 >= 4.0.0)

Versendet eine Email

string **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]])

This function is currently not available for PHP4 on WIN32.

XLIII. Informix functions

The Informix driver for Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x and IDS 2000 is implemented in "ifx.ec" and "php3_ifx.h" in the informix extension directory. IDS 7.x support is fairly complete, with full support for BYTE and TEXT columns. IUS 9.x support is partly finished: the new data types are there, but SLOB and CLOB support is still under construction.

Configuration notes: You need a version of ESQL/C to compile the PHP Informix driver. ESQL/C versions from 7.2x on should be OK. ESQL/C is now part of the Informix Client SDK.

Make sure that the "INFORMIXDIR" variable has been set, and that $INFORMIXDIR/bin is in your PATH before you run the "configure" script.

The configure script will autodetect the libraries and include directories, if you run "configure --with_informix=yes". You can override this detection by specifying "IFX_LIBDIR", "IFX_LIBS" and "IFX_INCDIR" in the environment. The configure script will also try to detect your Informix server version. It will set the "HAVE_IFX_IUS" conditional compilation variable if your Informix version >= 9.00.

Runtime considerations: Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to phpinfo() before you start testing. The phpinfo() output should list these environment variables. This is TRUE for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script.

The Informix shared libraries should also be available to the loader (check LD_LINBRARY_PATH or ld.so.conf/ldconfig).

Some notes on the use of BLOBs (TEXT and BYTE columns): BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id" for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with : "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with "ifx_create_blob();". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with ifx_update_blob().

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime :

configuration variable : ifx.textasvarchar

configuration variable : ifx.byteasvarchar

runtime functions :

ifx_textasvarchar(0) : use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0) : use blob id's for select queries with BYTE columns

ifx_textasvarchar(1) : return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1) : return BYTE columns as if they were VARCHAR columns, so that you

don’t need to use blob id’s for select queries.

configuration variable : ifx.blobinfile

runtime function :

ifx_blobinfile_mode(0) : return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1) : return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like : putenv(blobdir=tmpblob"); will ease the cleaning up of temp files accidentally left behind (their names all start with "blb").

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data: This can be set with the configuration variable

ifx.charasvarchar : if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

`NULL` values: The configuration variable ifx.nullformat (and the runtime function ifx_nullformat()) when set to `TRUE` will return `NULL` columns as the string "`NULL`", when set to `FALSE` they return the empty string. This allows you to discriminate between `NULL` columns and empty columns.

ifx_connect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Open Informix server connection

int **ifx_connect** ([string database [, string userid [, string password]]])

Returns a connection identifier on success, or `FALSE` on error.

ifx_connect() establishes a connection to an Informix server. All of the arguments are optional, and if they're missing, defaults are taken from values supplied in configuration file (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

In case a second call is made to **ifx_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling ifx_close().

See also ifx_pconnect(), and ifx_close().

Beispiel 1. Connect to a Informix database

```
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
```

ifx_pconnect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Open persistent Informix connection

int **ifx_pconnect** ([string database [, string userid [, string password]]])

Returns: A positive Informix persistent link identifier on success, or `FALSE` on error

ifx_pconnect() acts very much like ifx_connect() with two major differences.

This function behaves exactly like ifx_connect() when PHP is not running as an Apache module. First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ifx_close() will not close links established by **ifx_pconnect()**).

This type of links is therefore called 'persistent'.

See also: ifx_connect().

ifx_close (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Close Informix connection

int **ifx_close** ([int link_identifier])

Returns: always TRUE.

ifx_close() closes the link to an Informix database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by ifx_pconnect().

See also: ifx_connect(), and ifx_pconnect().

Beispiel 1. Closing a Informix connection

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

ifx_query (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Send Informix query

int **ifx_query** (string query [, int link_identifier [, int cursor_type [, mixed blobidarray]]])

Returns: A positive Informix result identifier on success, or FALSE on error.

A "result_id" resource used by other functions to retrieve the query results. Sets "affected_rows" for retrieval by the ifx_affected_rows() function.

ifx_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if ifx_connect() was called, and use it.

Executes *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together. Non-select queries are "execute immediate". IFX_SCROLL and IFX_HOLD are symbolic constants and as such shouldn't be between quotes. I you omit this parameter the cursor is a normal sequential cursor.

For either query type the number of (estimated or real) affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in an update query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_connect().

Beispiel 1. Show all rows of the "orders" table as a html table

```
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Beispiel 2. Insert some values into the "catalog" table

```
                   // create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                      // store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                      // launch query
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
  ... error ...
}
                     // free result id
ifx_free_result($res_id);
```

ifx_prepare (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Prepare an SQL-statement for execution

int **ifx_prepare** (string query, int conn_id [, int cursor_def, mixed blobidarray])

Returns a integer *result_id* for use by ifx_do(). Sets *affected_rows* for retrieval by the ifx_affected_rows() function.

Prepares *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together.

For either query type the estimated number of affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in the query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_do().

ifx_do (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Execute a previously prepared SQL-statement

int **ifx_do** (int result_id)

Returns TRUE on success, FALSE on error.

Executes a previously prepared query or opens a cursor for it.

Does NOT free *result_id* on error.

Also sets the real number of ifx_affected_rows() for non-select statements for retrieval by ifx_affected_rows()

See also: ifx_prepare(). There is a example.

ifx_error (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns error code of last Informix call

string **ifx_error** (void)

The Informix error codes (SQLSTATE & SQLCODE) formatted as follows :

x [SQLSTATE = aa bbb SQLCODE=cccc]

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed. The format of the string is "(char) [SQLSTATE=(two digits) (three digits) SQLCODE=(one digit)]". The first character can be '`  `' (space) (success), '`W`' (the statement caused some warning), '`E`' (an error happened when executing the statement) or '`N`' (the statement didn't return any data).

See also: ifx_errormsg()

ifx_errormsg (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Returns error message of last Informix call

string **ifx_errormsg** ([int errorcode])

Returns the Informix error message associated with the most recent Informix error, or, when the optional "*`errorcode`*" param is present, the error message corresponding to "*`errorcode`*".

See also: ifx_error()

```
printf("%s\n&lt;br>", ifx_errormsg(-201));
```

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Get number of rows affected by a query

int **ifx_affected_rows** (int result_id)

`result_id` is a valid result id returned by ifx_query() or ifx_prepare().

Returns the number of rows affected by a query associated with *`result_id`*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it. The database server can never return the actual number of rows that will be returned by a SELECT because it has not even begun fetching them at this stage (just after the "PREPARE" when the optimizer has determined the query plan).

Useful after ifx_prepare() to limit queries to reasonable result sets.

See also: ifx_num_rows()

Beispiel 1. Informix affected rows

```
$rid = ifx_prepare ("select * from emp
                     where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
```

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the contents of sqlca.sqlerrd[0..5] after a query

array **ifx_getsqlca** (int result_id)

result_id is a valid result id returned by ifx_query() or ifx_prepare().

Returns a pseudo-row (associative array) with sqlca.sqlerrd[0] ... sqlca.sqlerrd[5] after the query associated with *result_id*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For SELECTs the values are those saved after the PREPARE statement. This gives access to the *estimated* number of affected rows. The use of this function saves the overhead of executing a "select dbinfo('sqlca.sqlerrdx')" query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.

Beispiel 1. Retrieve Informix sqlca.sqlerrd[x] values

```
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n";
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Get row as enumerated array

array **ifx_fetch_row** (int result_id [, mixed position])

Returns an associative array that corresponds to the fetched row, or `FALSE` if there are no more rows.

Blob columns are returned as integer blob id values for use in ifx_get_blob() unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values. Returns `FALSE` on error

`result_id` is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

`position` is an optional parameter for a "fetch" operation on "scroll" cursors: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for SCROLL cursors.

ifx_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0, with the column name as key.

Subsequent calls to **ifx_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

Beispiel 1. Informix fetch rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                     $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Formats all rows of a query into a HTML table

int **ifx_htmltbl_result** (int result_id [, string html_table_options])

Returns the number of rows fetched or `FALSE` on error.

Formats all rows of the *`result_id`* query into a html table. The optional second argument is a string of <table> tag options

Beispiel 1. Informix results as HTML table

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                     $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do($rid) {
  ... error ...
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

List of Informix SQL fields

array **ifx_fieldtypes** (int result_id)

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *`result_id`*. Returns `FALSE` on error.

Beispiel 1. Fieldnames and SQL fieldtypes

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
```

```
}
```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

List of SQL fieldproperties

array **ifx_fieldproperties** (int result_id)

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *result_id*. Returns FALSE on error.

Returns the Informix SQL fieldproperties of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Beispiel 1. Informix SQL fieldproperties

```
$properties = ifx_fieldproperties ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_num_fields (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns the number of columns in the query

int **ifx_num_fields** (int result_id)

Returns the number of columns in query for *result_id* or FALSE on error

After preparing or executing a query, this call gives you the number of columns in the query.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Count the rows already fetched from a query

int **ifx_num_rows** (int result_id)

Gives the number of rows fetched so far for a query with *result_id* after a ifx_query() or ifx_do() query.

ifx_free_result (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Releases resources for the query

int **ifx_free_result** (int result_id)

Releases resources for the query associated with *result_id*. Returns FALSE on error.

ifx_create_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Creates an char object

int **ifx_create_char** (string param)

Creates an char object. *param* should be the char content.

ifx_free_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deletes the char object

int **ifx_free_char** (int bid)

Deletes the charobject for the given char object-id *bid*. Returns FALSE on error otherwise TRUE.

ifx_update_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Updates the content of the char object

int **ifx_update_char** (int bid, string content)

Updates the content of the char object for the given char object *bid*. *content* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_get_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Return the content of the char object

int **ifx_get_char** (int bid)

Returns the content of the char object for the given char object-id *bid*.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Creates an blob object

int **ifx_create_blob** (int type, int mode, string param)

Creates an blob object.

type: 1 = TEXT, 0 = BYTE

mode: 0 = blob-object holds the content in memory, 1 = blob-object holds the content in file.

param: if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.

Return FALSE on error, otherwise the new blob object-id.

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Duplicates the given blob object

int **ifx_copy_blob** (int bid)

Duplicates the given blob object. *bid* is the ID of the blob object.

Returns FALSE on error otherwise the new blob object-id.

ifx_free_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Deletes the blob object

int **ifx_free_blob** (int bid)

Deletes the blobobject for the given blob object-id *bid*. Returns FALSE on error otherwise TRUE.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Return the content of a blob object

int **ifx_get_blob** (int bid)

Returns the content of the blob object for the given blob object-id *bid*.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Updates the content of the blob object

ifx_update_blob (int bid, string content)

Updates the content of the blob object for the given blob object *bid*. *content* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Set the default blob mode for all select queries

void **ifx_blobinfile_mode** (int mode)

Set the default blob mode for all select queries. Mode "0" means save Byte-Blobs in memory, and mode "1" means save Byte-Blobs in a file.

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Set the default text mode

void **ifx_textasvarchar** (int mode)

Sets the default text mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Set the default byte mode

void **ifx_byteasvarchar** (int mode)

Sets the default byte mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_nullformat (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Sets the default return value on a fetch row

void **ifx_nullformat** (int mode)

Sets the default return value of a NULL-value on a fetch row. Mode "0" returns "", and mode "1" returns "NULL".

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Creates an slob object and opens it

int **ifxus_create_slob** (int mode)

Creates an slob object and opens it. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. You can also use constants named IFX_LO_RDONLY, IFX_LO_WRONLY etc. Return `FALSE` on error otherwise the new slob object-id.

ifxus_free_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Deletes the slob object

int **ifxus_free_slob** (int bid)

Deletes the slob object. *`bid`* is the Id of the slob object. Returns `FALSE` on error otherwise `TRUE`.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Deletes the slob object

int **ifxus_close_slob** (int bid)

Deletes the slob object on the given slob object-id *`bid`*. Return `FALSE` on error otherwise `TRUE`.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Opens an slob object

int **ifxus_open_slob** (long bid, int mode)

Opens an slob object. *`bid`* should be an existing slob id. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. Returns `FALSE` on error otherwise the new slob object-id.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Returns the current file or seek position

int **ifxus_tell_slob** (long bid)

Returns the current file or seek position of an open slob object *`bid`* should be an existing slob id. Return `FALSE` on error otherwise the seek position.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Sets the current file or seek position

int **ifxus_seek_slob** (long bid, int mode, long offset)

Sets the current file or seek position of an open slob object. *bid* should be an existing slob id. Modes: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END and *offset* is an byte offset. Return FALSE on error otherwise the seek position.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Reads nbytes of the slob object

int **ifxus_read_slob** (long bid, long nbytes)

Reads nbytes of the slob object. *bid* is a existing slob id and *nbytes* is the number of bytes read. Return FALSE on error otherwise the string.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Writes a string into the slob object

int **ifxus_write_slob** (long bid, string content)

Writes a string into the slob object. *bid* is a existing slob id and *content* the content to write. Return FALSE on error otherwise bytes written.

XLIV. InterBase-Funktionen

InterBase ist eine bekannte Datenbank, die von Borland/Inprise stammt. Weitere Informationen zu InterBase finden sie unter http://www.interbase.com/. Wichtig: InterBase befindet sich gerade auf dem Weg zum Open Source!

Anmerkung: Umfassende Unterstützung für InterBase 6 wurde in PHP 4.0 eingeführt.

ibase_connect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Öffnet eine Verbindung zu einer InterBase-Datenbank

int **ibase_connect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]])

Stellt eine Verbindung zu einem InterBase-Server her. Das *database*-Argument muss ein gültiger Pfad zu einer bestehenden Datenbank des Servers sein. Ist der DB-Server nicht lokal, muss ihm - in Abhängigkeit vom benutzten Protokoll - ein 'hostname:' (bei TCP/IP), '//hostname/' (bei NetBEUI) oder 'hostname@' (bei IPX/SPX) vorangestellt werden. Die Parameter *username* und *password* können auch über die PHP-Konfigurations-Anweisungen ibase.default_user und ibase.default_password gesetzt werden. *Charset* ist der reguläre Zeichensatz der Datenbank. Mit *buffers* geben sie die Anzahl der Datenbank-Cache-Buffer (serverseitig) an. Ist der Wert 0 oder nicht angegeben, benutzt der Server seine Default-Werte. Mittels *dialect* bestimmen sie den Default-Wert des anzuwendenden SQL-Dialekts für alle Anweisungen während einer Datenbank-Verbindung (default ist der Client-seitig höchst verfügbare Dialekt).

Wenn **ibase_connect()** mit den selben Argumenten erneut aufgerufen wird, wird keine neue Verbindung hergestellt, sondern die ID der bereits geöffneten Verbindung zurück gegeben. Die Verbindung zum InterBase-Server wird am Ende des jeweiligen Skripts automatisch beendet, es sei denn, es wird vorher die Funktion ibase_close() aufgerufen.

Beispiel 1. ibase_connect()-Beispiel:

```
<?php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Anmerkung: *Buffers* existiert seit PHP4-RC2.

Anmerkung: *Dialect* wurde in PHP4-RC2 hinzugefügt. Er funktioniert ab der InterBase-Version 6.

Anmerkung: *Role* existiert seit PHP4-RC2. Er funktioniert ab der InterBase-Version 5.

Siehe auch: ibase_pconnect().

ibase_pconnect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Erzeugt eine permanente Verbindung zu einer InterBase-Datenbank

int **ibase_connect** (string database [, string username [, string password [, string charset [, string role]]]])

ibase_pconnect() agiert ähnlich der Funktion ibase_connect() (mit zwei Ausnahmen). Erstens wird beim Verbindungs-Aufbau versucht, eine bereits bestehende permanente Verbindung zu finden, die mit den selben Parametern aufgebaut worden ist. Wird eine solche gefunden, wird statt eines neuen Verbindungs-Aufbaus die ID der bestehenden Verbindung zurück gegeben. Zweitens wird beim Beenden des Skripts die Verbindung nicht beendet. Statt dessen wird die Verbindung für den weiteren Gebrauch beibehalten (ibase_close() beendet keine Verbindung, die mittels **ibase_pconnect()** erzeugt wurde). Diese Art der Verbindung wird deshalb auch 'persistent' (permanent) genannt.

Siehe auch ibase_connect() hinsichtlich der Bedeutung der dieser Funktion übergebenen Parameter. Es sind die gleichen.

ibase_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schließt die Verbindung zu einer InterBase-Datenbank

int **ibase_close** ([int connection_id])

Beendet die Verbindung zu einer InterBase-DB, deren Verbindungs-ID mit der von ibase_connect() zurück gegebenen ID überein stimmt. Wird die Verbindungs-ID weg gelassen, wird die zuletzt hergestellte Verbindung beendet. Vorgenommene Transaktionen der Verbindung werden übertragen, andere werden zurück genommen.

ibase_query (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Führt eine Abfrage (Query) auf eine InterBase-DB aus

int **ibase_query** ([int link_identifier, string query [, int bind_args]])

Erzeugt eine Anfrage auf eine InterBase-DB, deren Ergebnis mit ibase_fetch_row(), ibase_fetch_object(), ibase_free_result() und ibase_free_query() weiter verarbeitet werden kann.

Anmerkung: Obwohl diese Funktion auch die Verwendung von Platzhaltern unterstützt, ist ihr keine besondere Bedeutung beizumessen. Für reale Abfragen und Beispiele sehen sie unter ibase_prepare() und ibase_execute() nach.

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liest einen Datensatz aus einer InterBase-Datenbank

array **ibase_fetch_row** (int result_identifier)

Gibt den nächsten Datensatz des per ibase_query() gewonnenen Zeigers in einem Array zurück.

ibase_fetch_object (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Liest einen Datensatz einer InterBase-Datenbank als Objekt ein

object **ibase_fetch_object** (int result_id)

Liest die Daten der per ibase_query() oder ibase_execute() gewonnenen *result_id* in ein Pseudo-Objekt ein.

```
<php
    $dbh  = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth  = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Siehe auch ibase_fetch_row().

ibase_free_result (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den Speicher eines Abfrage-Ergebnisses frei

int **ibase_free_result** (int result_identifier)

Gibt den durch den Aufruf von ibase_query() belegten Speicher wieder frei.

ibase_prepare (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Vorbereitung einer Abfrage für den folgenden Gebrauch von Parameter-Platzhaltern und für die eigentliche Ausführung.

int **ibase_prepare** ([int link_identifier, string query])

Vorbereitung einer Abfrage für den folgenden Gebrauch von Parameter-Platzhaltern und für die eigentliche Ausführung (per ibase_execute()).

ibase_execute (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ausführen einer vorbereiteten Abfrage

int **ibase_execute** (int query [, int bind_args])

Führt eine vorher per ibase_prepare() vorbereitete Abfrage aus. Dies ist effizienter als der Gebrauch von ibase_query(), sofern sie mehrere ähnliche Abfragen durchführen wollen, bei denen sich immer nur einige Parameter ändern.

```
<?php
    $updates = array(
        1 => 'Erik',
        5 => 'Phillip',
        7 => 'Ludwig'
    );

    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");

    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_free_query (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den Speicher einer vorbereiteten Abfrage wieder frei

int **ibase_free_query** (int query)

Gibt den Speicher einer vorbereiteten Abfrage (siehe ibase_prepare()) wieder frei.

ibase_timefmt (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Bestimmt das Format von Zeitstempel-, Datums- und Zeit-Feldern, die von einer Abfrage zurück gegeben werden

int **ibase_timefmt** (string format [, int columntype])

Bestimmt das Format von Zeitstempel-, Datums- und Zeit-Feldern, die durch eine Abfrage zurück gegeben werden. Intern werden diese Felder durch die C-Funktion strftime() formatiert. Deshalb finden sie in der dazu bestehenden Dokumentation genaue Hinweise über das entsprechende String-Format. Der optionale Parameter `columntype` kann eine der Konstanten IBASE_TIMESTAMP, IBASE_DATE oder IBASE_TIME sein. Fehlt er, wird per default von IBASE_TIMESTAMP ausgegangen (wegen der Abwärts-Kompatibilität).

```
<?php
    // InterBase 6 Zeit-Typ-Felder werden im
```

```
    // Format '05 Stunden 37 Minuten' zurück gegeben.
    ibase_timefmt("%H Stunden %M Minuten", IBASE_TIME);
?>
```

Sie können über die PHP-Konfigurations-Anweisungen ibase.timestampformat, ibase.dateformat und ibase.timeformat den default-Wert anpassen.

Anmerkung: *`Columntype`* wurde in PHP 4.0 hinzu gefügt. Er hat erst ab InterBase-Version 6 eine Bedeutung.

Anmerkung: Eine Abwärts-Inkompatibilität wurde in PHP 4.0 behoben, indem die Konfigurations-Anweisung ibase.timeformat umbenannt wurde in ibase.timestampformat und die Anweisungen ibase.dateformat und ibase.timeformat hinzu gefügt wurden. Dadurch entsprechen die Namen eher ihrer eigentlichen Funktionalität.

ibase_num_fields (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ermittelt die Anzahl der Felder einer Ergebnis-Liste

int **ibase_num_fields** (int result_id)

Der zurück gegebene Integer-Wert stellt die Anzahl der Felder einer Ergebnis-Liste dar.

```
<?php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);

    if (ibase_num_fields($sth) > 0) {
        while ($row = ibase_fetch_object ($sth)) {
            print $row->email . "\n";
        }
    } else {
        die ("Keine Ergebnisse für ihre Abfrage");
    }

    ibase_close ($dbh);
?>
```

Siehe auch: **ibase_field_info()**.

Anmerkung: ibase_num_fields() funktioniert z.Zt. nicht bei PHP 4.

XLV. Ingres II Funktionen

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

Mit diesen Funktionen können Sie auf einen Ingres II Datenbankserver zugreifen.

Damit Ihnen diese Funktionen zur Verfügung stehen, müssen Sie php mit der Ingres-Unterstützung kompilieren, indem Sie die Option `--with-ingres` angeben. Ausserdem brauchen Sie die Open API Bibliothek sowie header Dateien, die in Ingres II eingebunden werden. Falls die II_SYSTEM Umgebungsvariable nicht korrekt gesetzt ist, müssen Sie vielleicht `--with-ingres=DIR` benutzen, um Ihr Inges Installationsverzeichnis anzugeben.

Wenn Sie diese Erweiterung mit Apache benutzen, und Ihr Apache fährt nicht hoch, sondern bringt die Fehlermeldung: "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0", dann vergewissern Sie sich, dass die Umgebungsvariable II_SYSTEM korrekt gesetzt ist. Wenn Sie im Startskript für den Apache "export II_SYSTEM="/home/ingres/II" angeben, gerade bevor httpd gestartet wird, sollte dieses Problem gelöst sein.

Anmerkung: Falls Sie schon andere PHP Erweiterungen zum Zugriff auf andere Datenbankserver benutzt haben, sollten Sie beachten, dass Sie mit Ingres keine konkurrierenden Abfragen und/oder Transaktionen über eine Verbindung durchführen können. Darum finden Sie in dieser Erweiterung keine Ergebnis- oder Transaktions-Handler. Die Ergebnisse einer Abfrage müssen verarbeitet werden, bevor eine neue Abfrage an den Datenbankserver geschickt wird und eine Transaktion muss commited oder zurückgesetzt (roll-back) werden, bevor eine neue Transaktion geöffnet wird (die Transaktion wird automatisch geöffnet, wenn die erste Abfrage gesendet wird).

ingres_connect (PHP 4 >= 4.0.2)

Eine Verbindung zu einem Ingres II Datenbankserver öffnen

resource **ingres_connect** ([string database [, string username [, string password]]])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Gibt bei Erfolg eine Ingres II Verbindungskennung zurück, oder `FALSE`, wenn ein Fehler auftritt.

ingres_connect() öffnet eine Verbindung zu dem Ingres Datenbankserver, der im *`database`* Argument angegeben ist und die Syntax *`[node_id::]dbname[/svr_class]`* hat.

Wenn einige Parameter fehlen, benutzt **ingres_connect()** die Werte, die in der `php.ini` für *`ingres.default_database`*, *`ingres.default_user`* und *`ingres.default_password`* eingetragen sind.

Die Verbindung wird geschlossen, wenn das Skript zuende ist oder wenn die Funktion ingres_close() mit der aktuellen Verbindungskennung aufgerufen wird.

Alle anderen Ingres-Funktionen benutzen standardmässig die zuletzt geöffnete Verbindung, so dass Sie den Rückgabewert nur dann speichern müssen, wenn Sie mehr als eine Verbindung zur gleichen Zeit benutzen.

Beispiel 1. ingres_connect() Beispiel

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Verbindung fehlgeschlagen");
    print ("Verbindungsaufbau erfolgreich");
    ingres_close ($link);
?>
```

Beispiel 2. ingres_connect() Beispiel mit der aktuellen Verbindungskennung

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Verbindung fehlgeschlagen");
    print ("Verbindungsaufbau erfolgreich");
    ingres_close ();
?>
```

Siehe auch ingres_pconnect() und ingres_close().

ingres_pconnect (PHP 4 >= 4.0.2)

Öffnet eine persistente Verbindung zu einem Ingres II Datenbankserver

resource **ingres_pconnect** ([string database [, string username [, string password]]])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Gibt bei Erfolg eine Ingres II Verbindungskennung zurück oder oder `FALSE`, wenn ein Fehler aufgetreten ist.

Zu den Details über die Parameter siehe ingres_connect() und die Beispiele. Es gibt nur zwei Unterschiede zwischen den Funktionen **ingres_pconnect()** und ingres_connect(): Zum ersten: beim Verbindungsaufbau schaut die Funktion zuerst, ob schon eine geöffnete persistente Verbindung besteht, die mit denselben Parametern geöffnet wurde. Falls eine gefunden wird, wird ein Identifizierer zurückgegeben, anstatt eine neue Verbindung zu öffnen. Zum zweiten: Die Verbindung zum Ingres Server wird nicht geschlossen wenn die Ausführung des Skripts beendet ist, sondern die Verbindung bleibt zum weiteren Gebrauch offen (ingres_close() schliesst die Verbindungen, die mit **ingres_pconnect()** geöffnet wurden nicht). Darum wird diese Art der Verbindung 'persistent' genannt.

Siehe auch ingres_connect() und ingres_close().

ingres_close (PHP 4 >= 4.0.2)

Schliesst eine Ingres II Datenbankverbindung

bool **ingres_close** ([resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Gibt bei Erfolg `TRUE` zurück, bei einem Fehler `FALSE`.

ingres_close() schliesst die Verbindung zu dem Ingres Datenbankserver, der mit der übergebenen Verbindunkskennung verknüpft ist. Falls der Parameter *`link`* nicht angegeben wurde, wird die zuletzt geöffnete Verbindung geschlossen.

ingres_close() ist normalerweise nicht nötig, weil es keine persistenten Verbindungen schliesst und weil alle nicht-persistenten Verbindungen automatisch zum Ende des Skripts geschlossen werden.

Siehe auch ingres_connect() und ingres_pconnect().

ingres_query (PHP 4 >= 4.0.2)

Sendet eine SQL Abfrage an den Ingres II Server

bool **ingres_query** (string query [, resource link])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Gibt bei Erfolg `TRUE` zurück und bei einem Fehler `FALSE`.

ingres_query() sendet die übergebene *`query`* an den Ingres Server. Diese Abfrage muss ein gültiger SQL-String sein (Siehe die Ingres Dokumentation).

Die Abfrage wird zu einem Teil der aktuell offenen Transaktion. Falls keine Transaktion geöffnet ist, öffnet **ingres_query()** eine neue Transaktion. Um diese zu schliessen, können Sie entweder ingres_commit() aufrufen, um die gemachten Änderungen in die Datenbank zu schreiben, oder Sie rufen ingres_rollback() auf, um diese Änderungen zrückzusetzen. Wenn das Skript beendet wird, wird jede offene Transaktion zurückgesetzt (indem die Funktion ingres_rollback() aufgerufen wird). Sie können auch die Funktion ingres_autocommit() benutzen, bevor Sie eine neue Transaktion öffnen, damit werden alle einzelnen SQL-Abfrage sofort gesendet.

Einige SQL Abfragen können mit dieser Funktion nicht gesendet werden:

- close (siehe ingres_close())
- commit (siehe ingres_commit())
- connect (siehe ingres_connect())
- disconnect (siehe ingres_close())
- get dbevent
- prepare to commit
- rollback (siehe ingres_rollback())
- savepoint
- set autocommit (siehe ingres_autocommit())
- und alle Abfragen, die Cursors benutzen, werden nicht unterstützt

Beispiel 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
```

```
?>
```

Siehe auch ingres_fetch_array(), ingres_fetch_object(), ingres_fetch_row(), ingres_commit(), ingres_rollback() und ingres_autocommit().

ingres_num_rows (PHP 4 >= 4.0.2)

Gibt die Anzahl der Datensätze der Ergebnismenge zurück oder die Anzahl der von der letzten Abfrage betroffenen Datensätze

int **ingres_num_rows** ([resource link])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Bei delete-, insert- or update-queries, gibt **ingres_num_rows()** die Anzahl der durch diese Abfragen betroffenen Datensätze zurück. Bei den anderen Abfragen gibt **ingres_num_rows()** die Anzahl der Datensätze in der Ergebnismenge zurück.

Anmerkung: Diese Funktion dient hauptsächlich dazu, die Anzahl der veränderten Datensätze in einer Datenbank festzustellen. Wenn diese Funktion aufgerufen wird, bevor ingres_fetch_array(), ingres_fetch_object() oder ingres_fetch_row() aufgerufen wurde, löscht der Server die Ergebnisdaten so dass das Skript sie nicht auslesen kann.

Sie sollten stattdessen die Daten der Ergebnismenge mit einer dieser fetch-Funktionen in einer Schleife auslesen, solange, bis keine Datensätze mehr zum Lesen da sind und damit die Schleifenbedingung FALSE wird.

Siehe auch ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_num_fields (PHP 4 >= 4.0.2)

Gibt die Anzahl der Felder der letzten Abfrage zurück

int **ingres_num_fields** ([resource link])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_num_fields() liefert die Anzahl der Felder in der Ergebnismenge zurück, die vom Ingres Server nach einem Aufruf von ingres_query() zurückgegeben wurde.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_field_name (PHP 4 >= 4.0.2)

Holt den Namen eines Feldes in einer Ergebnismenge

string **ingres_field_name** (int index [, resource link])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_field_name() gibt den Namen eines Feldes in einer Ergebnismenge zurück oder `FALSE`, falls ein Fehler aufgetreten ist.

`index` gibt die Nummer des Feldes an und muss zwischen 1 und dem Wert, der von ingres_num_fields() zurückgegeben wird, liegen.

Siehe auch ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_field_type (PHP 4 >= 4.0.2)

Gibt den Typ eines Feldes in einer Ergebnismenge zurück

string **ingres_field_type** (int index [, resource link])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_field_type() gibt den Typ eines Feldes in einer Ergebnismenge zurück oder `FALSE`, wenn ein Fehler aufgetreten ist. Beispiele von Rückgabewerten sind: "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Einige dieser Typen können, in Abhängigkeit von der Feldlänge, mehrere

SQL Typen umfassen. (siehe ingres_field_length()). Beispielsweise kann "IIAPI_FLT_TYPE" entweder float4 oder float8 sein. Genauere Informationen darüber finden Sie in dem Ingres/OpenAPI User Guide - Appendix C.

index ist die Feldnummer und muss zwischen 1 und dem Wert liegen, den die Funktion ingres_num_fields() zurückgibt.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_field_nullable (PHP 4 >= 4.0.2)

Prüft, ob für ein Feld NULL-Werte zugelassen sind

bool **ingres_field_nullable** (int index [, resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_field_nullable() gibt `TRUE` zurück, falls in dieser Spalte `NULL`-Werte zugelasssen sind und `FALSE`, falls das nicht erlaubt ist.

index ist die Feldnummer und muss zwischen 1 und dem Wert liegen, den die Funktion ingres_num_fields() zurückgibt.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_field_length (PHP 4 >= 4.0.2)

Gibt die Länge eines Feldes zurück

int **ingres_field_length** (int index [, resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_field_length() gibt die Länge eines Feldes zurück. Das ist die Anzahl der Bytes, die der Server zum Speichern des Feldes belegt. Für genauere Informationen schauen sie in den Ingres/OpenAPI User Guide - Appendix C.

index ist die Feldnummer und muss zwischen 1 und dem Wert liegen, den die Funktion ingres_num_fields() zurückgibt.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_field_precision (PHP 4 >= 4.0.2)

Gibt die Genauigkeit eines Feldes zurück

int **ingres_field_precision** (int index [, resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_field_precision() gibt die Genauigkeit eines Feldes zurück. Dieser Rückgabewert wird nur für Dezimalzahlen, Fliesskommazahlen und monetäre SQL Datentypen benutzt. Wenn Sie es genauer wissen wollen, schauen Sie sich den Ingres/OpenAPI User Guide - Appendix C an.

`index` ist die Feldnummer und muss zwischen 1 und dem Wert liegen, den die Funktion ingres_num_fields() zurückgibt.

Siehe auch ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_field_scale (PHP 4 >= 4.0.2)

Gibt den Skalar eines Feldes zurück

int **ingres_field_scale** (int index [, resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_field_scale() gibt den Skalar eines Feldes zurück.Dieser Rückgabewert wird nur für Dezimalzahlen, Fliesskommazahlen und monetäre SQL Datentypen benutzt. Für nähere Informationen schauen Sie sich den Ingres/OpenAPI User Guide - Appendix C an.

`index` ist die Feldnummer und muss zwischen 1 und dem Wert liegen, den die Funktion ingres_num_fields() zurückgibt.

Siehe auch ingres_query(), ingres_fetch_array(), ingres_fetch_object() und ingres_fetch_row().

ingres_fetch_array (PHP 4 >= 4.0.2)

Liest eine Zeile der Ergebnismenge in ein Array ein

array **ingres_fetch_array** ([int result_type [, resource link]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_fetch_array() Gibt ein Array zurück, das die die entsprechende Zeile der Ergebnismenge enthält oder `FALSE` falls das die Ergebnismenge keine Zeilen mehr enthält.

Diese Funktion ist eine erweiterte Version der Funktion ingres_fetch_row(). Zusätzlich zur Speicherung der Daten unter numerischen Indizes werden die Daten auch unter assoziativen Indizes im Ergebnisarray gespeichert, in dem die Feldnamen als Schlüssel benutzt werden.

Falls eine oder mehrere Spalten der Ergebnismenge dieselben Feldnamen haben, bekommt die letzte Spalte die Priorität. Um auf die anderen Spalten zuzugreifen, müssen sie entweder den numerischen Index der Spalte benutzen oder Aliase für die Spalten machen.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

`result_type` kann INGRES_NUM für ein numerisches Array sein oder INGRES_ASSOC für ein assoziatives Array, oder INGRES_BOTH (Standard).

Von der Geschwindigkeit her ist diese Funktion identisch zu ingres_fetch_object(), und fast so schnell wie ingres_fetch_row() (der Unterschied ist zu vernachlässigen).

Beispiel 1. ingres_fetch_array() Beispiel

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # benutzt ein assoziatives Array
    echo $row["fullname"];
    echo $row[1];          # benutzt ein numerisches Array
    echo $row[2];
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_object() und ingres_fetch_row().

ingres_fetch_row (PHP 4 >= 4.0.2)

Holt eine Zeile aus einer Ergebnismenge in ein numerisches Array

array **ingres_fetch_row** ([resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_fetch_row() gibt ein Array zurück, das die entsprechende Zeile der Ergebnismenge enthält oder `FALSE`, falls es keine Zeilen mehr in der Ergebnismenge gibt. Jede Spalte ist unter einem Array-Offset, beginnend mit 1, gespeichert.

Aufeinanderfolgende Aufrufe von **ingres_fetch_row()** würden immer die nächste Zeile der Ergebnismenge zurückgeben oder `FALSE`, wenn es keine Zeilen mehr gibt.

Beispiel 1. ingres_fetch_row() Beispiel

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

Siehe auch ingres_num_fields(), ingres_query(), ingres_fetch_array() und ingres_fetch_object().

ingres_fetch_object (PHP 4 >= 4.0.2)

Holt eine Zeile als Objekt

object **ingres_fetch_object** ([int result_type [, resource link]])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_fetch_object() Gibt ein Objekt zurück, das mit der geholten Zeile einer Ergebnismenge korrespondiert oder `FALSE`, wenn es keine Zeilen mehr in der Egebnismenge gibt.

Diese Funktion ist ähnlich zu ingres_fetch_array(), mit einem Unterschied - ein Objekt wird anstelle eines Arrays zurückgegeben. Indirekt heisst das, dass Sie die Daten nur über ihre Feldnamen ansprechen können und nicht über ihre Offsets (Zahlen sind unzulässige Namen für Objekteigenschaften).

Das optionale Argument *result_type* ist eine Konstante, die die folgenden Werte annehmen kann: INGRES_ASSOC, INGRES_NUM und INGRES_BOTH.

Diese Funktion ist von der Geschwindigkeit her identisch zu ingres_fetch_array(), und fast so schnell wie ingres_fetch_row() (der Unterschied ist zu vernachlässigen).

Beispiel 1. ingres_fetch_object() Beispiel

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

Siehe auch ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_array() und ingres_fetch_row().

ingres_rollback (PHP 4 >= 4.0.2)

Setzt eine Transaktion zurück

bool **ingres_rollback** ([resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_rollback() setzt die aktuell geöffnete Transaktion zurück und macht alle Änderungen, die in der Transaktion bereits durchgeführt wurden, wieder rückgängig.

Diese Funktion schliesst die Transaktion. Eine Neue Transaktion kann geöffnet werden, wenn man mit ingres_query() eine neue Abfrage an den Datenbankserver schickt.

Siehe auch ingres_query(), ingres_commit() und ingres_autocommit().

ingres_commit (PHP 4 >= 4.0.2)

Schliesst eine Transaktion ab

bool **ingres_commit** ([resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_commit() schliesst die aktuell offene Transaktion ab, indem alle Veränderungen in die Datenbank übernommen werden.

Dadurch wird die Transaktion beendet. Eine neue Transaktion kann geöffnet werden, indem man eine neue Abfrage mit ingres_query() an den Datenbankserver schickt.

Sie können auch den Server automatisch nach jeder Abfrage einen Commit ausführen lassen, wenn sie die Funktion ingres_autocommit() aufrufen, bevor Sie die Transaktion öffnen.

Siehe auch ingres_query(), ingres_rollback() und ingres_autocommit().

ingres_autocommit (PHP 4 >= 4.0.2)

Schaltet den autocommit-Modus an oder aus

bool **ingres_autocommit** ([resource link])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ingres_autocommit() wird vor dem Öffnen einer Transaktion aufgerufen (vor dem ersten Aufruf der Funktion ingres_query() oder gleich nach einem Aufruf von ingres_rollback() oder **ingres_autocommit()**) um in den "autocommit" Modus des Servers ein- oder auszuschalten (beim Beginn eines Skripts ist der autocommit-Modus ausgeschaltet).

Wenn der autocommit-Modus eingeschaltet ist, wird jede Abfrage automatisch sofort vom Server commited, so als würde die Funktion ingres_commit() nach jedem Aufruf von ingres_query() aufgerufen werden.

Siehe auch ingres_query(), ingres_rollback() und ingres_commit().

XLVI. IRC Gateway Funktionen

Was ist ircg?

IRCG ermöglicht die Entwicklung schneller und skalierbarer Webchats die über einen privaten oder öffentlichen Webserver kommunizieren.

Plattformen

IRCG kann auf folgenden Betriebssystemen verwendet werden:

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64

Benötigte Software

Um die IRCG-Funktionen zu verwenden, muss folgende Software installiert sein:

1. IRCG-Library (http://www.schumann.cx/ircg/) von Sascha Schumann.
2. SGI Static Threads Library (http://sourceforge.net/projects/state-threads/)
3. thttpd (http://www.acme.com/software/thttpd/) webserver

Installation

Eine ausführliche Installationsanleitung befindet sich hier (http://lxr.php.net/source/php4/ext/ircg/README.txt).

ircg_pconnect (PHP 4 >= 4.0.4)

Verbindung zum IRC-Server aufbauen

resource **ircg_pconnect** (string username [, string server_ip [, int server_port [, string msg_format [, array ctcp_messages [, array user_settings]]]]])

ircg_pconnect() Stellt eine Verbindung zum einem IRC Server her, und liefert eine Verbindunskennung.

Der Parameter *username* muss angeben werden, er legt den Benutzernamen auf dem Server fest. *server_ip* und *server_port* sind optional. Werden diese nicht angegben wird für *server_ip* der Wert 127.0.0.1 (localhost) und für *server_port* der Wert 6667 gesetzt.

> **Anmerkung:** Der Parameter *server_ip* muss aus einer IP-Adresse in numerischer Form bestehen.

Die Ausgabe der IRC Meldungen und Ereignisse wird mit einem formatierten String, der vor dem Aufruf mit der Funktion ircg_register_format_messages() im Parameter *msg_format* definiert.

ctcp_messages

user_settings

Siehe auch: ircg_disconnect(), ircg_is_conn_alive(), ircg_register_format_messages().

ircg_fetch_error_msg (PHP 4 >= 4.1.0)

Liefert den Fehler des letzten ircg Funktionsaufrufes.

array **ircg_fetch_error_msg** (resource connection)

ircg_fetch_error_msg() Liefet den Fehler der Verbindung *connection* der letzten aufgerufenen ircg Funktion zurück.

> **Anmerkung:** Die Fehlernummer wird im ersten Element des Arrays zurückgeliefert, der Fehlertext im zweiten Parameter.

Beispiel 1. ircg_fetch_error_msg() Beispiel

```
if (!ircg_join ($id, "#php")){
    $error = ircg_fetch_error_msg($id);
    print ("Can’t join channel #php. Errorcode: $error[0] Description: $error[1]");
}
```

ircg_set_current (PHP 4 >= 4.0.4)

Setzt die Verbindung für die Ausgabe der IRC-Server Meldungen

boolean **ircg_set_current** (resource connection)

Setzt die Verbindung für die Ausgabe. Alle Meldungen des verbundenen IRC-Servers, der über den Parameter `connection` spezifiziert wurde, werden über die in der Funktion ircg_register_format_messages() angegebenen Formatierungen ausgegeben.

Siehe auch: ircg_register_format_messages() and ircg_lookup_format_messages().

ircg_join (PHP 4 >= 4.0.4)

Einen Channel auf dem verbundenen IRC-Server betreten

boolean **ircg_join** (resource connection, string channel)

Betritt den Channel `channel` auf dem IRC-Server `connection`.

Siehe auch:

ircg_part (PHP 4 >= 4.0.4)

Einen Channel auf dem IRC-Server verlassen

boolean **ircg_part** (resource connection, string channel)

Verlässt den Channel `channel` auf dem IRC-Server `connection`.

Siehe auch:

ircg_msg (PHP 4 >= 4.0.4)

Sendet eine Nachricht an einen Channel oder Benutzer

boolean **ircg_msg** (resource connection, string recipient, string message [, boolean suppress])

ircg_msg() sendet eine Nachricht an einen Channel oder Benutzer auf dem IRC-Server `connection`. Der Parameter `recipient`, beginnend mit einem # oder & sendet die Nachricht an einen Channel, andere Angaben werden als Benutzername interpretiert.

Wird der optionale Parameter `suppress` auf `TRUE` gesetzt, wird die Ausgabe der Nachricht auf der eigenen Verbindung unterbunden.

ircg_notice (PHP 4 >= 4.0.5)

Sendet eine Notiz zu einem anderen Benutzer

boolean **ircg_notice** (resource connection, string , string message)

Sendet eine Nachricht *message* zum angegebenen Benutzer *nick* auf dem verbunden IRC-Server *connection*. Der genaue Unterschied zwischen einer Nachricht und einer Notiz ist dem IRC-Manual zu entnehmen.

Siehe auch:

ircg_nick (PHP 4 >= 4.0.5)

ändert den Benutzernamen auf dem IRC-Server

boolean **ircg_nick** (resource connection, string nick)

Ändert den Benutzernamen auf dem IRC-Server *connection* auf den neuen Benutzernamen *nick*.

liefert bei erfolgreicher Änderung `TRUE` oder `FALSE` falls der name nicht geändert werden konnte.

ircg_topic (PHP 4 >= 4.0.5)

Setzt das Thema für einen Channel

boolean **ircg_topic** (resource connection, string channel, string new_topic)

Ändert das Thema (Topic) für den Channel *channel* des verbundenen IRC-Servers *connection* in *new_topic*.

ircg_channel_mode (PHP 4 >= 4.0.5)

Setzt die Channel Attribute für einen Benutzer

boolean **ircg_channel_mode** (resource connection, string channel, string mode_spec, string nick)

Setzt die Attribute *mode_spec* in einem Channel *channel* auf dem verbundenen Server *connection* für den Benutzer *nick*.

Die Attribute werden gesetzt oder gelöscht durch Angabe eines Zeichen verbunden mit einem Plus oder Minuszeichen. Ein Operator-Attribut beispielsweise wird mittels '+o' gesetzt oder durch '-o' entfernt.

Siehe auch:

ircg_html_encode (PHP 4 >= 4.0.5)

Wandelt einen HTML string für die Ausgabe um

boolean **ircg_html_encode** (string html_string)

Wandelt den HTML-String `html_string` für die Ausgabe um. Dies ist beispielsweise sinnvoll, wenn in einem Channel über HTML oder Beispiele diskutiert wird.

Siehe auch:

ircg_whois (PHP 4 >= 4.0.5)

Liefert die Information über einen bestimmten Benutzer

boolean **ircg_whois** (resource connection, string nick)

Weist den verbunden IRC-Server `connection` an, Informationen über den Benutzer `nick` anzuzeigen.

Siehe auch:

ircg_kick (PHP 4 >= 4.0.5)

Entfernt einen Benutzer aus einem Channel

boolean **ircg_kick** (resource connection, string channel, string nick, string reason)

Entfernt den Benutzer `nick` vom Channel `channel` auf dem verbundenen IRC-Server `connection`. `reason` gibt den Grund an, warum der User aus dem Channel geworfen wird.

ircg_ignore_add (PHP 4 >= 4.0.5)

Fügt einen Benutzer an die Liste der zu ignorierenden Personen an.

boolean **ircg_ignore_add** (resource connection, string nick)

Die Function fügt den Benutzer `nick` zur persönlichen Liste der ignorierten Benutzer auf dem verbundenen IRC-Server `connection` hinzu. Dadurch werden von diesem Benutzer keine Nachrichten mehr angezeigt.

Siehe auch: ircg_ignore_del() .

ircg_ignore_del (PHP 4 >= 4.0.5)

Entfernt einen Benutzer von der Liste ignorierter Personen

boolean **ircg_ignore_del** (resource connection, string nick)

Diese Funktion entfernt den Benutzer `nick` von der Liste der ignorierten Personen auf dem verbunden IRC-Server `connection`.

Siehe auch: ircg_ignore_add().

ircg_disconnect (PHP 4 >= 4.0.4)

Beendet die Verbindung zum IRC-Server

boolean **ircg_disconnect** (resource connection, string reason)

ircg_disconnect() beendet die Verbindung *connection* zu einem IRC-Server, die mittels der Funktion ircg-pconnect() hergestellt wurde

Siehe auch: ircg_pconnect().

ircg_is_conn_alive (PHP 4 >= 4.0.5)

Check connection status

boolean **ircg_is_conn_alive** (resource connection)

ircg_is_conn_alive() gibt den Wert `TRUE` zurück, wenn die Verbindung *connection* zum IRC-Server besteht, oder `FALSE` falls keine gültige Verbindung mehr besteht.

ircg_lookup_format_messages (PHP 4 >= 4.0.5)

Wählt ein Format für die Ausgabe von IRC-Nachrichten und Ereignissen aus.

boolean **ircg_lookup_format_messages** (string name)

Wählt ein über die Funktion **ircg_register_format()** definiertes Format *string* zur Formatierung der Ausgabe von IRC-Server Nachrichten und Ereignissen aus. Unter dem Namen `ircg` steht ein Standardformat zur Verfügung.

Siehe auch: ircg_register_format_messages()

ircg_register_format_messages (PHP 4 >= 4.0.5)

Registriert die Ausgabeformatierungen für Nachrichten und Ereignisse vom IRC-Server

boolean **ircg_register_format_messages** (string name, array messages)

Mit der Funktion **ircg_register_format_messages()** kann die Formatierung der Ausgabe von IRC-Servermeldungen und -ereignissen festgelegt werden. Dabei ist es möglich mehrere Formate (Layer) zu definieren, die mit der Funktion ircg_lookup_format_messages() ausgewählt werden können.

Die Funktionen für die Ausgabe werden in dem Array *messages* in folgender Reihenfolge übergeben:

- Nachricht an den Channel
- Erhalten einer privaten Nachricht

- Private Nachricht gesendet
- Ein Benutzer verlässt den Channel
- Ein Benutzer betritt den Channel
- Ein Benutzer wurde aus dem Channel geworfen (gekicked)
- Thema (Topic) des Channels hat sich geändert
- Fehler
- schwerer Fehler
- Ende der Benutzerliste (wird beim Betreten eines Channels gesendet)
- Meldung beim eigenen Betreten eines Channels
- Ein Benutzer ändert seinen Namen
- Ein Benutzer beendet die Verbindung zum IRC-Server
- Eine grössere Anzahl von Benutzern betritt den Channel (z.B. Bei netsplit/join)
- Ein Benutzer aus der Liste bei massenhaftem Betreten eines Channels
- Ende des massenhaften Betretens eines Channels
- Benutzerinformation
- Serverinformation
- Information über Inaktivität
- Information über den Channel
- Ende Meldung bei Information über Channel (Listenende)
- Benachrichtigung über Änderung des voice-Attributs
- Benachrichtung über Änderung des Operator-Attributs
- für den Channel gesperrter Benutzer
- Ende der Liste gesperrter Benutzer

- %f - von
- %t - an
- %c - channel
- %r - vollständige Nachricht
- %m - umgewandelte Nachricht
- %j - für Javascript umgewandelte Nachricht

- 1 - mod encode
- 2 - decodierter Benutzername

Siehe auch: ircg_lookup_format_messages().

ircg_set_on_die (PHP 4 CVS only)

Setzt eine Aktion bei Abbruch der Verbindung

bool **ircg_set_on_die** (int connection, string host, int port, string data)

Im Falle eines Verbindungsabruchs der Verbindung *connection* sendet **ircg_set_on_die()** auf einem Irc-Server, der über *host* und *port* (Hinweis, der Parameter *host* muss eine gültige IPv4-Adresse enthalten) angegeben wird, die Zeichenkette *data*.

ircg_set_file (PHP 4 CVS only)

Legt eine Logdatei an

bool **ircg_set_file** (int connection, string path)

Die Funktion **ircg_set_file()** legt eine Logdatei *path* für die Verbindung *connection* an, in der alle Ausgaben geloggt werden. **ircg_set_file()** liefert bei Auftreten eines Fehler FALSE zurück, ansonsten TRUE.

ircg_get_username (PHP 4 >= 4.1.0)

Liefert den Benutzernamen zurück

string **ircg_get_username** (int connection)

Die Funktion **ircg_get_username()** liefert den aktuellen Benutzernamen für die Verbindung *connection* zurück. Ist die Verbindung ungültig, oder wurde die Verbindung abgebrochen, liefert **ircg_get_username()** FALSE zurück.

ircg_nickname_escape (PHP 4 >= 4.0.6)

Formatiert einen Benutzernamen in einen gültigen IRC-Namen

string **ircg_nickname_escape** (string nick)

Die Funktion **ircg_nickname_escape()** formatiert einen Benutzernamen *nick* in einen IRC konformes Format, und liefert den umgewandelten Namen zurück.

Siehe auch: ircg_nickname_unescape()

ircg_nickname_unescape (PHP 4 >= 4.0.6)

Decodiert einen IRC konformen Benutzernamen

string **ircg_nickname_unescape** (string nick)

Function **ircg_nickname_unescape()** liefert einen decodierten Benutzernamen zurück, der über den Parameter `nick` übergeben wurde.

Siehe auch: ircg_nickname_escape()

XLVII. Java

Es gibt zwei Wege, um die Welten von Java und PHP zu verbinden. Einerseits besteht die Möglichkeit PHP in eine Java Servlet Umgebung zu integrieren, andererseits kann man mit der Java Extension aus PHP heraus auf Java Klassen zugreifen. Die erste der beiden genannten Möglichkeit ist wesentlich stabiler und performanter als die zweitgenannte Lösung.

Die Java Erweiterung für PHP nutzt das Object Overloading von PHP 4 um auf Java Klassen zuzugreifen. Die Java Virtual Machine (JVM) wird hierbei über das Java Native Interface (JNI) erzeugt. Eine detaillierte Installationsbeschreibung entnehmen Sie bitte der Datei `php4/ext/java/README`.

Beispiel 1. Zugriff auf Java Klassen aus PHP heraus

```
<?php
  // In PHP Instanz der Java Klasse java.lang.System erzeugen
  $system = new Java('java.lang.System');

  // Zugriff auf die Objekteigenschaften
  print 'Java version='.$system->getProperty('java.version').' <br>';
  print 'Java vendor=' .$system->getProperty('java.vendor').'  <br>';
  print 'OS='.$system->getProperty('os.name').' '.
              $system->getProperty('os.version').' on '.
              $system->getProperty('os.arch').' <br>';

  // java.util.Date Beispiel
  $formatter = new Java('java.text.SimpleDateFormat',
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

  print $formatter->format(new Java('java.util.Date'));
?>
```

Beispiel 2. Zugriff auf Java AWT aus PHP heraus

```
<?php
  // Dieses Beispiel kann nur als CGI ausgeführt werden.

  $frame  = new Java('java.awt.Frame', 'PHP');
  $button = new Java('java.awt.Button', 'Hallo Java Welt!');

  $frame->add('North', $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;

  $thread = new Java('java.lang.Thread');
  $thread->sleep(10000);

  $frame->dispose();
?>
```

Notes:

- `new Java()` erzeugt eine Instanz der angegebenen Java Klasse, falls ein geeigneter Konstruktur gefunden wird. Für den Zugriff auf Klassen mit statischen Methoden benötigt man keinen Parameter.
- Beim Zugriff auf Mitglieder (Members) einer Instanz wird zunächst nach Bean Eigenschaften gesucht, danach erst nach öffentlichen (public) Eigenschaften. Mit anderen Worten: Es wird zuerst versucht werden, zum Beispiel `print $date.time` als `$date.getTime()` zu interpretieren, danach erst wird `$date.time` versucht werden.
- Der Zugriff auf statische und Instanzeigenschaften erfolgt mit der selben Syntax.
- Das Auslösen einer Java Exception resultiert in einer PHP Warnung und einem NULL Ergebnis. Folgende Funktionen ermöglichen den Zugriff auf die eigentlichen Exceptions:
  - java_last_exception_get()
  - java_last_exception_clear()
- Traditionell können Arrays und Hashes in PHP weitestgehend als gleichwertig angesehen werden. Beim Austausch dieser Datentypen zwischen PHP und Java ist zu beachten, dass Hashes in PHP nur mit Integer- und Stringwerten indiziert werden können und Arrays von primitiven Datentypen in Java nicht dünn besetzt sein dürfen. Ferner ist zu beachten, dass diese Datenstrukturen by-Value übergeben werden, was ungünstig in Bezug auf Speicher- und Zeitbedarf sein kann.

Das PHP 4 Servlet SAPI Modul baut auf dem von der Java Extension zur Verfügung gestellten Mechanismus auf, allerdings wird der PHP Prozessor hierbei von einer Servlet Engine, wie zum Beispiel Apache Jakarta / Tomcat, ausgeführt. Dies führt zu einer wesentlich höheren Stabilität und besserer Performance als der umgekehrte Weg, wie in die Java Extension an sich anbietet. Dies kommt daher, dass die Servlet Engine sich um das Pooling und die Wiederverwendung von Java Virtual Machines (JVMs) kümmert. Eine detaillierte Anleitung zur Integration von PHP in eine Servlet Engine entnehmen Sie bitte der Datei `php4/sapi/README`. Bemerkungen:

- Das Servlet SAPI Modul sollte zwar auf jeder Java Servlet Engine funktionieren, wurde aber bislang nur in Verbindung mit Apache Jakarta/Tomcat getestet. Meldungen von Problem - aber auch von Erfolgserlebnissen - bei der Verendung mit anderen Servlet Engines werden dankbar aufgenommen.
- PHP hat die Angewohnheit das aktuelle Arbeitsverzeichnis (CWD) zu ändern. sapi/servlet wird versuchen, den alten Zustand wieder herzustellen. Während PHP läuft kann es sein, dass die Servlet Engine kann Klassen aus dem CLASSPATH finden kann, die über relative Pfade angesprochen werden.

java_last_exception_clear (PHP 4 >= 4.0.2)

Letzte Java Exception freigeben

void **java_last_exception_clear** (void)

java_last_exception_get (PHP 4 >= 4.0.2)

Letzte Java Exception holen

exception **java_last_exception_get** (void)

Das nachstehende Beispiel demonstriert die Verarbeitung von Java Exceptions aus PHP heraus:

Beispiel 1. Verarbeitung von Java Exceptions

XLVIII. LDAP Funktionen

Einführung zu LDAP

LDAP steht für Lightweight Directory Access Protocol und ist ein Protokoll um auf "Directory Servers" - "Verzeichnis- Server" - zuzugreifen. Das Verzeichnis ist dabei eine spezielle Art einer Datenbank, das Informationen in einer Baumstruktur bereithält.

Das Konzept ist dabei ähnlich der Verzeichnisstruktur Ihrer Festplatte, mit der Ausnahme, daß in diesem Zusammenhang das Wurzelverzeichnis "Die Welt" ist und die Unterverzeichnisse der ersten Ebene "Länder" abbilden. Die weiteren Verzeichnisse unterhalb der Länderebene enthalten Einträge für Firmen, Organisationen oder Städte. Noch tiefer geschachtelt stehen Verzeichniseinträge für Menschen und vielleicht für Ausstattung oder Dokumente.

Um auf eine Datei in einem Unterverzeichnis auf Ihrer Festplatte zu verweisen, könnten Sie eine Schreibweise wie folgt verwenden

```
/usr/local/meine_daten/doku
```

Die Schrägstriche markieren jeden Teil in diesem Verweis und die Sequenz wird von links nach rechts gelesen.

Das entsprechende Gegenstück zu einer voll qualifizierten Dateireferenz ist in LDAP der "Eindeutige Name" ("distinguished name"), der Einfachheit halber als "dn" bezeichnet. Ein Beispiel für einen solchen dn könnte sein

```
cn=Hans Mustermann,ou=Buchhaltung,o=Meine Firma,c=DE
```

Das Komma markiert jeden Teil in diesem Verweis und die Sequenz wird von rechts nach links gelesen. Diesen dn würden Sie wie folgt lesen ...

```
Land = DE
Organisation (organisation) = Meine Firma
Organisationseinheit (organisational unit) = Buchhaltung
Einfacher Name (commonName) = Hans Mustermann
```

In der gleichen Weise, in der es keine allgemeinverbindlichen Regeln gibt, wie Sie die Verzeichnisstruktur Ihrer Festplatte organsieren, kann der Verwalter eines Verzeichnis-Servers jede beliebige Struktur implementieren, die für den Einsatzzweck sinnvoll ist. Trotzdem gibt es einige Konventionen die benutzt werden. Die Botschaft ist folgende: Sie können keinen Code schreiben um auf einen Verzeichnis-Server zuzugreifen, ohne etwas über dessen Struktur zu wissen. Genauso wenig können Sie eine Datenbank nutzen ohne Kenntnis darüber, was in derselben vorhanden ist.

Komplettes Code Beispiel

Informationen von einem Verzeichnis-Server beziehen für alle Einträge bei denen der Nachname mit

"S" beginnt. Darstellung eines Auszugs mit Name und Email-Adresse.

Beispiel 1. LDAP Such-Beispiel

```
<?php
    // Grundlegende Abfolge bei LDAP ist verbinden, binden, suchen,
    // interpretieren des Sucheergebnisses, Verbindung schließen

    echo "<h3>LDAP query Test</h3>";
    echo "Verbindung ...";
    $ds=ldap_connect("localhost");  // muss ein gültiger LDAP Server
                                    // sein!

    echo "Ergebnis der Verbindung: ".$ds."<p>";

    if ($ds) {
    echo "Bindung ...";
    $r=ldap_bind($ds);     // das ist ein "anonymer" bind,
                           // typischerweise nur Lese Zugriff
    echo "Ergebnis der Bindung ".$r."<p>";

    echo "Suche nach (sn=S*) ...";
    // Suchen des Nachnamen-Eintrags
    $sr=ldap_search($ds,"o=Meine Firma, c=DE", "sn=S*");
    echo "Ergebnis der Suche ".$sr."<p>";

    echo "Anzahl gefundenen Einträge ".ldap_count_entries($ds,$sr)."<p>";

    echo "Einträge holen ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Daten für ".$info["count"]." Items gefunden:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn ist: ". $info[$i]["dn"] ."<br>";
        echo "erster cn Eintrag: ". $info[$i]["cn"][0] ."<br>";
        echo "erster email Eintrag: ". $info[$i]["mail"][0] ."<p>";
    }

    echo "Verbindung schließen";
    ldap_close($ds);

} else {
    echo "<h4>Verbindung zu LDAP Server nicht möglich</h4>";
}
?>
```

Verwenden der PHP LDAP Aufrufe

Sie müssen sich die LDAP Client Bibliotheken herunterladen und übersetzen. Sie können entweder das ldap-3.3 Packet von der University of Michigan oder das Netscape Directory SDK 3.0

verwenden. Ebenso müssen Sie Ihr PHP mit LDAP Unterstützung aktiviert neu übersetzen bevor die PHP LDAP-Aufrufe funktionieren.

Bevor Sie die LDAP Aufrufe benutzen können, müssen Sie folgendes wissen...

- Den Namen oder die Adresse des Verzeichnis-Servers den Sie benutzen wollen
- Den "Basis dn" des Servers (den Teil des Weltverzeichnisses, das auf diesem Server eingerichtet ist, das könnte "o=Meine Firma,c=DE" sein.
- Brauchen Sie ein Passwort um auf den Server zuzugreifen? (Viele Server bieten Lesezugriff für einen "anonymen bind", verlangen aber ein Passwort für alles weitere)

Die typische Abfolge von LDAP Aufrufen in einer Anwendung folgen diesem Muster

```
ldap_connect()    // Aufbau einer Verbindung zum Server

connection to server
  |
ldap_bind()       // anonymes oder authentifiziertes "login"
  |
mache irgendwas, z.B. Verzeichnis durchsuchen oder aktualisieren
und stelle das Ergebnis dar
  |
ldap_close()      // "logout"
```

Mehr Informationen

Viele Informationen finden Sie unter

- Netscape (http://developer.netscape.com/tech/directory/)
- University of Michigan (http://www.umich.edu/~dirsvcs/ldap/index.html)
- OpenLDAP Project (http://www.openldap.org/)
- LDAP World (http://www.innosoft.com/ldapworld)

Das Netscape SDK enthält einen hilfreichen Programmer´s Guide im .html Format.

ldap_add (PHP 3, PHP 4 >= 4.0.0)

Einträge einem LDAP Verzeichnis hinzufügen

int **ldap_add** (int Verbindungs-Kennung, string dn, array eintrag)

Rückgabewert: Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Die **ldap_add()** Funktion wird benutzt um Einträge einem Verzeichnis hinzuzufügen. Die DN des Eintrags, der hinzugefügt werden soll wird durch *`dn`* bestimmt. Das Array *`eintrag`* bestimmt die Informationen des Eintrags. Die Werte in den Einträgen werden durch individuelle Merkmale indiziert. Im Fall von mehreren Werten für ein Merkmal werden diese durch die Verwendung von Zahlen indiziert, die bei 0 beginnen.

eintrag["merkmal1"] = wert
eintrag["merkmal2"][0] = wert1
eintrag["merkmal2"][1] = wert2

Beispiel 1. Komplettes Beispiel mit authentifiziertem bind

```
<?php
$ds=ldap_connect("localhost");  // Annahme: der LDAP Server befindet
                                // sich auf diesem Host

if ($ds) {
    // bind mit passendem dn für aktulisierenden Zugriff
    $r=ldap_bind($ds,"cn=root, o=Meine Firma, c=DE", "geheim");

    // Daten vorbereiten
    $info["cn"]="Hans Mustermann";
    $info["sn"]="Mustermann";
    $info["mail"]="MustermannH@firma.de";
    $info["objectclass"]="person";

    // hinzufügen der Daten zum Verzeichnis
    $r=ldap_add($ds, "cn=Hans Mustermann, o=Meine Firma, c=DE", $info);

    ldap_close($ds);
} else {
    echo "Verbindung zum LDAP Server nicht möglich!";
}
?>
```

ldap_bind (PHP 3, PHP 4 >= 4.0.0)

Bindung zu einem LDAP Verzeichnis

int **ldap_bind** (int Verbindungs-Kennung [, string bind_rdn [, string bind_kennwort]])

Bindung für ein LDAP Verzeichnis mit angegebenem RDN und Kennwort. Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Die **ldap_bind()** Funktion führt eine bind Operation auf einem Verzeichnis aus. Die Parameter *`bind_rdn`* und *`bind_kennwort`* sind optional, wenn diese fehlen wird ein anonymes bind versucht.

ldap_close (PHP 3, PHP 4 >= 4.0.0)

Verbindung zum LDAP Server schliessen

int **ldap_close** (int Verbindungs-Kennung)

Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Die **ldap_close()** Funktion schließt die Verbindung zum LDAP Server die zur *`Verbindungs-Kennung`* gehört.

Der Aufruf dieser Funktion ist intern identisch zu ldap_unbind(). Die LDAP API benutzt den Aufruf ldap_unbind(), so dass Sie vielleicht diese Funktion benutzen sollten, anstatt **ldap_close()** zu verwenden.

Anmerkung: Diese Funktion ist ein Alias für ldap_unbind().

ldap_compare (PHP 4 >= 4.0.2)

Vergleicht gefundenen Wert eines Merkmal in einem Eintrag, der durch Angabe von dn bestimmt wird.

int **ldap_compare** (int Verbindungs-Kennung, string dn, string merkmal, string wert)

Rückgabewert: wahr, wenn der *`wert`* gefunden wird, ansonsten `FALSE`. Tritt ein Fehler auf, wird -1 zurückgegeben.

ldap_compare() vergleicht den *`wert`* eines *`merkmal`* mit dem Wert des gleichen Merkmals in dem LDAP Verzeichniseintrag der durch *`dn`* bestimmt wurde.

Das folgende Beispiel zeigt, wie überprüft wird, ob ein gegebenes Kennwort mit dem Kennwort übereinstimmt oder nicht, das in dem durch DN bestimmten Eíntrag gespeichert ist.

Beispiel 1. Komplettes Beispiel einer Kennwort-Überprüfung

```
<?php

$ds=ldap_connect("localhost");  // Annahme: der LDAP Server befindet
                                // sich auf diesem Host

if ($ds) {

    // binden
```

```
    if(ldap_bind($ds)) {

        // Daten vobereiten
        $dn = "cn=Helga Olsen, ou=Meine Einheit, o=Meine Firma, c=FI";
        $value = "Geheimes Kennwort";
        $attr = "kennwort";

        // vergleiche die Werte
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Fehler: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Kennwort korrekt.";
        } elseif ($r === FALSE) {
            echo "Falsch geraten! Kennwort nicht korrekt.";
        }

    } else {
        echo "Keine Bindung zum LDAP Server möglich.";
    }

    ldap_close($ds);

} else {
    echo "Keine Verbindung zum LDAP Server möglich.";
}
?>
```

Warnung

ldap_compare() kann NICHT für Vergleiche von BINÄREN Werten benutzt werden!

Anmerkung: Diese Funktion wurde in 4.0.2 hinzugefügt.

ldap_connect (PHP 3, PHP 4 >= 4.0.0)

Verbindung zu einem LDAP Server

int **ldap_connect** ([string hostname [, int port]])

Rückgabewert: Eine positive Verbindungs-Kennung im Erfolgsfall, `FALSE` im Fehlerfall.

Die **ldap_connect()** Funktion stellt eine Verbindung zu einem LDAP Server auf einem bestimmten *`hostname`* und *`port`* her. Beide Argumente sind optional. Werden keine Argumente angegeben , wird die Verbindungs-Kennung einer schon geöffneten Verbindung zurückgeliefert. Wird nur der *`hostname`* angegeben, wird für den Port der Standarport 389 benutzt.

Bei Verwendung von OpenLDAP 2.x.x können Sie eine URL statt des Hostnamens angeben. Um LDAP mit SSL-Unterstützzung zu verwenden, übersetzen Sie OpenLDAP mit SSL-Unterstützung, PHP mit SSL und verwenden ldaps://Hostname/ als Host Argument. Das Port Argument wird bei der Verwendung von URLs nicht benutzt.

Anmerkung: URL und SSL Unterstützung wurden in 4.0.4 hinzugefügt

ldap_count_entries (PHP 3, PHP 4 >= 4.0.0)

Zählt die Anzahl der Einträge bei einer Suche

int **ldap_count_entries** (int Verbindungs-Kennung, int Ergebnis-Kennung)

Rückgabewert: Anzahl der Einträge im Ergebnis, `FALSE` im Fehlerfall.

Die **ldap_count_entries()** Funktion liefert die Anzahl der Einträge im gespeicherten Ergebnis einer vorher durchgeführten Suchoperation. Die *`Ergebnis-Kennung`* bezeichnet das interne LDAP Ergebnis.

ldap_delete (PHP 3, PHP 4 >= 4.0.0)

Löscht einen Eintrag aus einem Verzeichnis

int **ldap_delete** (int Verbindungs-Kennung, string dn)

Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Die **ldap_delete()** Funktion löscht einen einzelnen Eintrag aus einem LDAP Verzeichnis, das im *`dn`* angegeben wurde.

ldap_dn2ufn (PHP 3, PHP 4 >= 4.0.0)

Konvertiert DN in ein benutzerfreundliches Namensformat

string **ldap_dn2ufn** (string dn)

Die **ldap_dn2ufn()** Funktion wird verwendet um einen DN, angegeben mit *`dn`*, in einer benutzerfreundlicheren Form darzustellen, wobei die Typen-Namen wegfallen.

ldap_err2str (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Konvertiert eine LDAP Fehlernummer in einen Fehlertext

string **ldap_err2str** (int fehlernummer)

Rückgabewert: Fehlertext.

Diese Funktion liefert den Fehlertext der die *fehlernummer* erklärt. Während LDAP Fehlernummern standardisiert sind, liefern unterschiedliche Bibliotheken unterschiedliche oder sogar lokalspezifische Fehlermeldungen. Deshalb sollten Sie nie auf eine spezielle Fehlermeldung prüfen, sondern immer eine Fehlernummer verwenden.

Siehe auch ldap_errno() und ldap_error().

Beispiel 1. Aufzählung aller LDAP Fehlermeldungen

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Fehler $i: %s<br>\n", ldap_err2str($i));
  }
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Liefert die LDAP Fehlernummer des letzten LDAP Kommandos

int **ldap_errno** (int Verbindungs-Kennung)

Rückgabewert: LDAP Fehlernummer des letzten LDAP Kommandos für diese Verbindung.

Diese Funktion liefert die standardisierte Fehlernummer des zuletzt ausgeführten LDAP Kommandos für die angegebene Vebindungs-Kennung. Diese Nummer kann mit der Funktion ldap_err2str() in einen Fehlertext konvertiert werden.

Solange Sie die Stufe der Fehlerwarnungen in Ihrer php.ini nicht herabsetzen oder vor Ihre LDAP Komanndos ein @ (at) setzen um die Ausgabe von Fehlermeldungen zu unterdrücken, werden die aufgetretenen Fehler auch in Ihrer HTML Ausgabe zu sehen sein.

Beispiel 1. Erzeugen und Abfangen eines Fehlers

```
<?php
// Dieses Beispiel enhält einen Fehler, der abgefangen wird.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);

// Syntax Fehler im Filterausdruck (errno 87),
// müsste "objectclass=*" heißen, um zu funktionieren.

$res =  @ldap_search($ld, "o=Meine Firma, c=DE", "objectclass");
if (!$res) {
    printf("LDAP-Fehlernummer: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Fehler: %s<br>\n", ldap_error($ld));
    die("Argh!<br>\n");
}
```

```
$info = ldap_get_entries($ld, $res);
printf("%d Übereinstimmende Einträge.<br>\n", $info["count"]);
?>
```

Siehe auch ldap_err2str() und ldap_error().

ldap_error (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Liefert die LDAP Fehlermeldung des letzten LDAP Kommandos

string **ldap_error** (int Verbindungs-Kennung)

Rückgabe: Fehlertext

Diese Funktion liefert den erklärenden Fehlertext für einen Fehler, der vom zuletzt ausgeführten LDAP Kommando mit der angegebenen *Verbindungs-Kennung* erzeugt wurde. Während LDAP Fehlernummern standardisiert sind, liefern unterschiedliche Bibliotheken unterschiedliche oder sogar lokalspezifische Fehlermeldungen. Deshalb sollten Sie nie auf eine spezielle Fehlermeldung prüfen, sondern immer eine Fehlernummer verwenden.

Solange Sie die Stufe der Fehlerwarnungen in Ihrer `php.ini` nicht herabsetzen oder vor Ihre LDAP Komanndos ein `@` (at) setzen, um die Ausgabe von Fehlermeldungen zu unterdrücken, werden die aufgetretenen Fehler auch in Ihrer HTML Ausgabe zu sehen sein.

Siehe auch ldap_err2str() und ldap_errno().

ldap_explode_dn (PHP 3, PHP 4 >= 4.0.0)

Aufteilung eines DN in seine Bestandteile

array **ldap_explode_dn** (string dn, int mit_merkmal)

Die **ldap_explode_dn()** Funktion wird benutzt um einen von ldap_get_dn() gelieferten DN aufzuteilen und in dessen Bestandteile zu zerlegen. Jeder Teil wird als Relative Distinguished Name, RDN, bezeichnet. **ldap_explode_dn()** liefert einen Array mit allen diesen Bestandteilen. Der Parameter *mit_merkmal* wird benutzt, um zu bestimmen, ob die RDNs nur mit Werten oder zusätlich mit ihren Merkmalen angefragt werden. Um RDNs mit Merkmalen (z.B. Merkmal = Wert Format) zu erhalten, setzen Sie *mit_merkmal* auf 0, um nur Werte zu erhalten auf 1.

ldap_first_attribute (PHP 3, PHP 4 >= 4.0.0)

Liefert das erste Merkmal

string **ldap_first_attribute** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung, int ber-Kennung)

Rückgabewert: im Erfolgsfall das erste Merkmal des Eintrags, im Fehlerfall `FALSE`.

Ähnlich wie Einträge lesen, werden Merkmale eines bestimmten Eintrags eines nach dem anderen gelesen. Die **ldap_first_attribute()** Funktion liefert das erste Merkmal eines Eintrags der durch die *Ergebnis-Eintrag-Kennung* angegeben wurde. Durch erfolgreichen Aufruf von ldap_next_attribute() erhalten Sie die restlichen Merkmale. Der Parameter *ber-Kennung* ist die Kennung eines internen Speicherort-Zeiger. Die Übergabe erfolgt per Referenz. Die gleiche *ber-Kennung* wird an die Funktion ldap_next_attribute() weitergegeben, die diesen Zeiger dann verändert.

Siehe auch ldap_get_attributes()

ldap_first_entry (PHP 3, PHP 4 >= 4.0.0)

Liefert die Kennung des ersten Ergebnisses

int **ldap_first_entry** (int Verbindungs-Kennung, int Ergebnis-Kennung)

Rückgabewert: im Erfolgsfall die Ergebnis-Eintrag-Kennung für den ersten Eintrag, `FALSE` im Fehlerfall.

Einträge eines LDAP Ergebnisses werden unter Verwendung der Funktionen **ldap_first_entry()** und ldap_next_entry() sequentiell gelesen. Die **ldap_first_entry()** Funktion gibt die Eintrag-Kennung für den ersten Eintrag im Ergebnis zurück. Diese Eintrag-Kennung wird dann an die Funktion ldap_next_entry() weitergegeben, um sukzessiv die Einträge des Ergebnisses zu erhalten.

Siehe auch ldap_get_entries().

ldap_free_result (PHP 3, PHP 4 >= 4.0.0)

Gibt den belegten Speicher wieder frei

int **ldap_free_result** (int Ergebnis-Kennung)

Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Um die Ergebnisse zu speichern, wird intern Speicher belegt. Diesen Speicher können Sie mit der Funktion **ldap_free_result()** mir Angabe der entsprechenden *Ergebnis-Kennung* wieder freigeben. Typischerweise wird nach Beendigung des Skripts wird der gesamte Ergebnis-Speicher automatisch freigegeben.

Falls Ihr Skript sukzessive Suchoperationen durchführt, die große Ergebnismengen liefern, können Sie **ldap_free_result()** aufrufen, um den Speicherverbrauch zur Laufzeit niedrig zu halten.

ldap_get_attributes (PHP 3, PHP 4 >= 4.0.0)

Liefert Merkmale eines Suchergebnis-Eintrags

array **ldap_get_attributes** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung)

Rückgabewert: im Erfolgsfall wird die gesamte Information eines Eintrags in einem multidimensionalen Array zurückgegeben; im Fehlerfall `FALSE`.

Die **ldap_get_attributes()** Funktion wird verwendet um das Lesen der Merkmale und Werte eines Eintrags im Suchergebnis zu vereinfachen. Der Rückgabewert ist ein multidimensionales Array aus Merkmalen und Werten.

Nachdem Sie einen bestimmten Eintrag in dem Verzeichnis lokalisiert haben, können Sie mit dieser Funktion herausfinden, welche Informationen für diesen Eintrag verfügbar sind. Diese Funktion würden Sie in einer Anwendung verwenden, die Verzeichniseinträge "durchstöbert" und/oder wo Sie die Struktur der Verzeichniseinträge nicht kennen. In vielen Anwendungen suchen Sie nach einem bestimmten Merkmal wie einer email-Adresse oder einem Nachnamen und interessieren sich nicht dafür, welche anderen Daten verfügbar sind.

return_value["count"] = Anzahl der Merkmale des Eintrags
return_value[0] = das erste Merkmal
return_value[n] = das n-te Merkmal

return_value["merkmal"]["count"] = Anzahl der Werte des Merkmals
return_value["merkmal"][0] = der erste Wert des Merkmals
return_value["merkmal"][i] = der i-te Wert des Merkmals

Beispiel 1. Anzeigen der Liste der Merkmale für einen ausgewählten Verzeichnis-Eintrag

```
// $ds die Verbindungs-Kennung für das Verzeichnis

// $sr ein gültiges Suchergebnis eines früheren Aufrufs einer der ldap
// Verzeichnis Suchfunktionen


$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"]." Merkmale für diesen Eintrag:<p>";

for ($i=0; $i<$attrs["count"]; $i++)
    echo $attrs[$i]."<br>";
```

Siehe auch ldap_first_attribute() und ldap_next_attribute()

ldap_get_dn (PHP 3, PHP 4 >= 4.0.0)

Liefert den DN eines Ergebnis-Eintrags

string **ldap_get_dn** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung)

Rückgabewert: im Erfolgsfall der DN des Ergebnis-Eintrags, im Fehlerfall `FALSE`.

Die **ldap_get_dn()** Funktion wird verwendet, um den DN eines Eintrags im Ergebnis herauszufinden.

ldap_get_entries (PHP 3, PHP 4 >= 4.0.0)

Liefert alle Ergebnis-Einträge

array **ldap_get_entries** (int Verbindungs-Kennung, int Ergebnis-Kennung)

Rückgabewert: im Erfolgsfall wird die gesamte Information des Ergebnisses in einem multidimensionalen Array zurückgegeben; im Fehlerfall `FALSE`.

Die **ldap_get_entries()** Funktion wird verwendet um das Lesen mehrfacher Einträge des Ergebnisses, angegeben mit *`Ergebnis-Kennung`* zu vereinfachen und danach werden die Merkmale und mehfachen Werte gelesen. Durch einen Funktionsaufruf wird die gesamte Information in einem multidimensionalen Array zurückgegeben. Die Struktur dieses Array sieht wie folgt aus.

Der Merkmal-Index wird in Kleinbuchstaben umgewandelt. (Verzeichnis-Server unterscheiden nicht zwischen Groß-/Kleinschreibung hinsichtlich der Merkmake, aber der Unterschied ist wichtig, wenn diese Merkmale als Array-Indizes benutzt werden.)

```
return_value["count"] = Anzahl der Einträge im Ergebnis
return_value[0] : bezieht sich auf Einzelheiten des ersten Eintrags

return_value[i]["dn"] =  DN des i-ten Eintrags im Ergebnis

return_value[i]["count"] = Anzahl der Merkmale im i-ten Eintrag
return_value[i][j]       = das j-te Merkmal des i-ten Eintrags im Ergebnis

return_value[i]["merkmal"]["count"] = Anzahl der Werte für das
                                      Merkmal im i-ten Eintrag
return_value[i]["merkmal"][j]       = der j-te Wert des Merkmals im i-ten
                                      Eintrag
```

Siehe auch ldap_first_entry() und ldap_next_entry()

ldap_get_option (PHP 4 >= 4.0.4)

Liefert den aktuellen Wert für eine gegebene Option

bool **ldap_get_option** (int Verbindungs-Kennung, int option, mixed retval)

Setzt *`retval`* auf den Wert der gewählten Option. Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Der Parameter *`option`* kann einer der folgenden sein: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. Beschrieben sind diese unter draft-ietf-ldapext-ldap-c- api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt)

Anmerkung: Diese Funktion steht nur zur Verfügung, wenn Sie OpenLDAP 2.x.x ODER Netscape Directory SDK x.x benutzen. Wurde in PHP 4.0.4 hinzugefügt.

Beispiel 1. Prüfen der Protokollversion

```
// $ds eine gültige Verbindungs-Kennung für einen Verzeichnis-Server

if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "Sie benutzen die Protokollversion $version";
else
    echo "Protokollversion konnte nicht bestimmt werden";
```

Siehe auch ldap_set_option().

ldap_get_values (PHP 3, PHP 4 >= 4.0.0)

Liefert alle Werte eines Ergebnis-Eintrags

array **ldap_get_values** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung, string merkmal)

Rückgabewert: im Erfolgsfall ein Array der Werte des Merkmals, `FALSE` im Fehlerfall.

Die **ldap_get_values()** Funktion wird benutzt um alle Werte des Merkmals eines Eintrags im Ergebnis zu lesen. Der Eintrag wird durch *`Ergebnis-Eintrag-Kennung`* bestimmt. Die Anzahl der Werte kann über die Indexierung von "count" des resultierenden Arrays bestimmt werden. Der Zugriff auf einzelne Werte erfolgt durch einen ganzzahligen Index im Array. Der erste Indexwert ist 0.

Da diese Funktion eine *`Ergebnis-Eintrag-Kennung`* benötigt, muss ein Aufruf einer der LDAP Suchfunktionen und einer der Funktionen, die einen einzelnen Eintrag liefern, vorausgehen.

Ihre Anwendung wird entweder hart codiert sein, um bestimmte Merkmale (wie z.B.: "Nachname" oder "Mail") zu suchen, oder Sie müssen die Funktion ldap_get_attributes() benutzen, um herauszufinden, welche Merkmale für einen gegebenen Eintrag vorhanden sind.

LDAP erlaubt mehr als einen Eintrag für ein Merkmal, so daß es z.B. sein kann, dass der Verzeichnis-Eintrag einer Person mehrere Email-Adressen enthält, die alle mit dem Merkmal "mail" bezeichnet sind.

return_value["count"] = Anzahl der Werte eines Merkmals
return_value[0] = der erste Wert des Merkmals
return_value[i] = der i-te Wert des Merkmals

Beispiel 1. Liste aller Werte des Merkmals "mail" für einen Verzeichnis- Eintrag

```
// $ds gültige Verbindungs-Kennung für einen Verzeichnis-Server

// $sr gültiges Suchergebnis eines früheren Aufrufs einer der ldap
//     Verzeichnis Suchfunktionen

// $entry gültige Eintrag-Kennung eines früheren Aufrufs einer
//        der Funktionen, die einen Verzeichnis-Eintrag zurückgeben

$values = ldap_get_values($ds, $entry,"mail");

echo $values["count"]." Email-Adressen für diesen Eintrag.<p>";

for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
```

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert alle binären Werte eines Ergebnis-Eintrags

array **ldap_get_values_len** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung, string merkmal)

Rückgabewert: im Erfolgsfall ein Array der Werte des Merkmals, `FALSE` im Fehlerfall.

Die **ldap_get_values_len()** Funktion wird benutzt um alle Werte des Merkmals eines Eintrags im Ergebnis zu lesen. Der Eintrag wird durch *`Ergebnis-Eintrag-Kennung`* bestimmt. Die Anzahl der Werte kann über die Indexierung von "count" des resultierenden Arrays bestimmt werden. Der Zugriff auf einzelne Werte erfolgt durch einen ganzzahligen Index im Array. Der erste Indexwert ist 0.

Diese Funktion wird exakt gleich benutzt wie die Funktion ldap_get_values() mit der Aussnahme, dass hier binäre Daten statt Textdaten behandelt werden.

Anmerkung: Diese Funktion wurde in 4.0 hinzugefügt.

ldap_list (PHP 3, PHP 4 >= 4.0.0)

Einstufige Suche

int **ldap_list** (int Verbindungs-Kennung, string basis_dn, string filter [, array merkmale [, int nur_merkmmale [, int größenbegrenzung [, int zeitbegrenzung [, int deref]]]]])

Rückgabewert: im Erfolgsfall eine Such-Ergebnis-Kennung,im Fehlerfall `FALSE`.

Die **ldap_list()** Funktion führt die Suche für einen gegebenen *filter* im Verzeichnis mit der Reichweite von LDAP_SCOPE_ONELEVEL durch.

LDAP_SCOPE_ONELEVEL bedeutet, dass die Suche nur Informationen liefert, die direkt unterhalb der Stufe liegen, die durch die Angabe des *basis_dn* im Auruf bestimmt wird. (Das entspricht der Eingabe von "ls", die eine Liste von Dateien und Verzeichnissen des aktuellen Arbeitsverzeichnisses anzeigt.)

Der Aufruf dieser Funktion nimmt 5 optionale Parameter entgegen. Siehe die Anmerkungen zu ldap_search().

> **Anmerkung:** Diese wahlfreien Parameter wurden in 4.0.2 hinzugefügt: *attrsonly*, *größenbegrenzung*, *zeitbegrenzung*, *deref*.

Beispiel 1. Erzeugung einer Liste aller Organisationseinheiten einer Organisation

```
// $ds gültige Verbindungs-Kennung für einen Verzeichnis-Server

$basis_dn = "o=Meine Firma, c=DE";
$nur_dieses = array("ou");

$sr=ldap_list($ds, $basis_dn, "ou=*", $nur_dieses);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

> **Anmerkung:** Seit der Version 4.0.5 ist es außerdem möglich parallele Suchen durchzuführen. Für Details siehe ldap_search().

ldap_modify (PHP 3, PHP 4 >= 4.0.0)

Verändern eines LDAP-Eintrags

int **ldap_modify** (int Verbindungs-Kennung, string dn, array eintrag)

Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Die **ldap_modify()** Funktion wird verwendet, um bestehende Einträge in einem LDAP-Verzeichnis zu ändern. Die Struktur des Eintrags ist die gleiche wie bei ldap_add().

ldap_mod_add (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Hinzufügen von Merkmalswerten zu aktuellen Merkmalen

int **ldap_mod_add** (int Verbindungs-Kennung, string dn, array eintrag)

Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

Diese Funktion fügt Merkmal(e) zum angegebenen *dn* hinzu. Die Änderung wird auf der Merkmalsebene durchgeführt im Gegensatz zur Objektebene. Das Hinzufügen auf der Objektebene wird durch die Funktion ldap_add() geleistet.

ldap_mod_del (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Löschen von Merkmalswerten aktueller Merkmale

int **ldap_mod_del** (int Verbindungs-Kennung, string dn, array eintrag)

Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

Diese Funktion löscht ein oder mehrere Merkmale vom angegebenen *dn*. Die Änderung wird auf der Merkmalsebene durchgeführt im Gegensatz zur Objektebene. Das Löschen auf Objektebene wird durch die Funktion ldap_delete() geleistet.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Ersetzen von Merkmalswerten mit neuen Merkmalswerten

int **ldap_mod_replace** (int Verbindungs-Kennung, string dn, array eintrag)

Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

Diese Funktion ersetzt ein oder mehrere Merkmale vom angegebenen *dn*. Die Änderung wird auf der Merkmalsebene durchgeführt im Gegensatz zur Objektebene. Änderungen auf der Objektebene werden mit der Funktion ldap_modify() ausgeführt.

ldap_next_attribute (PHP 3, PHP 4 >= 4.0.0)

Liefert das nächste Merkmal im Ergebnis

string **ldap_next_attribute** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung, int ber_Kennung)

Rückgabewert: im Erfolgsfall das nächste Merkmal eines Eintrags, im Fehlerfall FALSE.

Die **ldap_next_attribute()** Funktion wird aufgerufen, um die Merkmale eines Eintrags zu erhalten. Der interne Status des Zeigers wird durch die *ber_Kennung* bestimmt. Diese wird mittels Referenz an die Funktion weitergegeben. Der erste Aufruf von **ldap_next_attribute()** erfolgt mit der *Ergebnis-Eintrag-Kennung*, die von ldap_first_attribute() zurückgegeben wurde.

Siehe auch ldap_get_attributes()

ldap_next_entry (PHP 3, PHP 4 >= 4.0.0)

Liefert den nächsten Eintrag des Ergebnisses

int **ldap_next_entry** (int Verbindungs-Kennung, int Ergebnis-Eintrag-Kennung)

Rückgabewert: Im Erfolgsfall wird die Eintrags-Kennung für den nächsten Eintrag des Ergebnisses zurückgegeben. Das Lesen der Einträge wird mit ldap_first_entry() gestartet. Wenn keine weiteren Einträge im Ergebnis vorhanden sind wird `FALSE` zurückgegeben.

Die **ldap_next_entry()** Funktion wird verwendet, um die Einträge zu erhalten, die im Ergebnis gespeichert sind. Wiederholtes aufrufen von **ldap_next_entry()** liefert einen Eintrag nach dem anderen, solange bis kein Eintrag mehr vorliegt. Der erste Aufruf von **ldap_next_entry()** erfolgt nach dem Aufruf von ldap_first_entry() mit der *`Ergebnis-Eintrag-Kennung`*, die von ldap_first_entry() zurückgegeben wurde.

Siehe auch ldap_get_entries()

ldap_read (PHP 3, PHP 4 >= 4.0.0)

Lesen eines Eintrags

int **ldap_read** (int Verbindungs-Kennung, string basis_dn, string filter [, array merkmale [, int nur_werte [, int größenbegrenzung [, int zeitbegrenzung [, int deref]]]]])

Rückgabewert: Im Erfolgsfall eine Such-Ergebnis-Kennung, `FALSE` im Fehlerfall.

Die **ldap_read()** Funktion führt die Suche für einen gegebenen *`filter`* im Verzeichnis mit der Reichweite von LDAP_SCOPE_BASE durch. Das entpricht dem Lesen eines Eintrags in einem Verzeichnis.

Ein leerer Filter ist nicht erlaubt. Wenn Sie wirklich alle Informationen für einen Eintrag erhalten möchten, müssen Sie einen Filter der Art "objectClass=*" verwenden. Kennen Sie die Typen der Einträge die auf dem Verzeichnis-Server benutzt werden, können Sie einen passenden Filter wie z.B."objectClass=inetOrgPerson" verwenden.

Der Aufruf dieser Funktion nimmt 5 optionale Argumente entgegen. Siehe Anmerkungen zu ldap_search().

> **Anmerkung:** Diese wahlfreien Argumente wurden in 4.0.2 hinzugefügt: *`nur_werte`*, *`größenbegrenzung`*, *`zeitbegrenzung`*, *`deref`*.
>
> Seit der Version 4.0.5 ist es außerdem möglich parallele Suchen durchzuführen. Für Details siehe ldap_search().

ldap_rename (PHP 4 >= 4.0.5)

Verändert den Namen eines Eintrags

bool **ldap_rename** (int Verbindungs-Kennung, string dn, string neuerrdn, string neueeltern, bool löschealtenrdn)

Der Eintrag, der durch *dn* gegeben ist, wird umbenannt/gelöscht. Der neue RDN wird durch *neuerrdn* bestimmt, der neue Eltern-/übergeordnete Eintrag wird durch *neueeltern* bestimmt. Wenn das Argument *löschealtenrdn* TRUE ist, wird der alte Wert/die alten Werte von RDN enfernt, andernfalls wird der alte Wert/die alten Werte als nicht-eindeutige(r) (non- distinguished) Werte des Eintrags beibehalten. Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

Anmerkung: Diese Funktion arbeitet zum gegenwärtigen Zeitpunkt nur mit LDAPv3. Sie müssen ldap_set_option() vorher benutzen um für eine Bindung LDAPv3 nutzen zu können. Diese Funktion steht nur zur Verfügung, wenn Sie OpenLDAP 2.x.x ODER Netscape Directory SDK x.x benutzen. Wurde in PHP 4.0.5 hinzugefügt.

ldap_search (PHP 3, PHP 4 >= 4.0.0)

Suche im LDAP Baum

int **ldap_search** (int Verbindungs-Kennung, string base_dn, string filter [, array merkmale [, int nur_werte [, int größenbegrenzung [, int zeitbegrenzung [, int deref]]]]])

Rückgabewert: eine Such-Ergebnis-Kennung im Erfolgsfall, FALSE im Fehlerfall.

Die **ldap_search()** Funktion führt die Suche für einen gegebenen Filter im Verzeichnis mit der Reichweite von LDAP_SCOPE_SUBTREE durch. Das ist äquivalent zu einer Suche im ganzen Verzeichnis.*base_dn* legt den Basis DN für das Verzeichnis fest.

Der optionale vierte Parameter kann benutzt werden, um die Rückgabewerte des Servers so einzuschränken, dass nur die tatsächlich benötigten Merkmale und ihre zugehörigen Werte in der Ergebnismenge enthalten sind. Dieses Vorgehen ist um einiges effizienter als die standardmäßige Vorgehensweise (diese liefert alle Merkmale und alle zugehörigen Werte). Aus diesem Grund ist die Angabe des vierten Parameters als gute Praxis zu empfehlen.

Der vierte Parameter ist ein Standard PHP Array aus Strings der benötigten Merkmale, z.B. array("mail","sn","cn"). Beachten Sie, dass der "dn" immer zurückgeliefert wird, ohne Rücksicht darauf, welche Merkmalstypen angefragt wurden.

Beachten Sie weiterhin, dass manche Verzeichnis-Server so konfiguriert sind, dass sie nicht mehr als eine vorbestimmte Anzahl an Einträgen zurückliefern. Sollte dies der Fall sein, zeigt Ihnen der Server an, dass nur eine Teilmenge des Ergebnisses zurückgeliefert wurde. Diesen Hinweis erhalten Sie auch, wenn Sie den sechsten Parameter *größenbegrenzung* mit angegeben hatten, um die Anzahl der angefragten Einträge einzuschränken.

Der fünfte Parameter *nur_werte* sollte auf 1 gesetzt werden, wenn Sie nur Merkmalstypen erhalten möchten. Wenn der Wert auf 0 steht, erhalten Sie sowohl Merkmalstypen als auch Merkmalswerte. Das ist das Standard Verhalten.

Mit dem sechsten Parameter *größenbegrenzung* ist es möglich, die Anzahl der Einträge, die Sie erhalten, zu begrenzen. Wenn Sie diesen Wert auf 0 setzen, bedeutet dies keine Beschränkung der Ergegnismenge. ANMERKUNG: Dieser Parameter kann eine serverseitig gesetzte Beschränkung NICHT überschreiben. Sie haben nur die Möglichkeit die Beschränkung noch weiter herabzusetzen.

Der siebte Parameter *zeitbegrenzung* legt die Zahl in Sekunden fest, die auf die Suche verwendet wird. Wenn Sie diesen Wert auf 0 setzen, bedeutet dies keine Beschränkung der Zeit.

ANMERKUNG: Dieser Parameter kann eine serverseitig gesetzte Zeitbegrenzung NICHT überschreiben. Sie haben nur die Möglichkeit die Beschränkung noch weiter herabzusetzen.

Der achte Parameter *deref* gibt an, wie Aliase während einer Suche behandelt werden. Wert kann einer der folgenden sein:

- LDAP_DEREF_NEVER - (Standard) Aliase werden nie aufgelöst.
- LDAP_DEREF_SEARCHING - Aliase sollen während der Suche aufgelöst werden, aber nicht dann, wenn das Basisobjekt der Suche ermittelt wird.
- LDAP_DEREF_FINDING - Aliase sollen aufgelöst werden, wenn das Basisobjekt ermiitelt wird, aber nicht während der Suche.
- LDAP_DEREF_ALWAYS - Aliase sollen immer aufgelöst werden.

Anmerkung: Diese optionalen Parameter wurden in 4.0.2 hinzugefügt: *nur_werte*, *größenbegrenzung*, *zeitbegrenzung*, *deref*.

Der Suchfilter kann einfach oder kompliziert sein, wenn sie boolsche Operatoren in dem Format verwenden, wie in der LDAP Dokumentation beschrieben (siehe Netscape Directory SDK (http://developer.netscape.com/docs/manuals/directory/41/ag/find.htm) für die vollständige Information über Filter).

Das untenstehende Beispiel liefert die Organisationseinheit, den Familiennamen, den Vornamen und die Email-Addresse aller Personen in "Meine Firma" deren Familien- oder Vorname die Zeichenkette $person enhält. In diesem Beispiel wird ein boolscher Filter verwendet, um den Server zu veranlassen, nach Informationen in mehr als einem Merkmal zu suchen.

Beispiel 1. LDAP Suche

```
// $ds gültige Verbindungs-Kennung für einen Verzeichnis-Server

// $person ein Teil oder der vollständige Name einer Person, z.B. "Jo"

$dn = "o=Meine Firma, c=DE";
$filter="(|(sn=$person*)(vorname=$person*))";
$justthese = array( "ou", "sn", "vorname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." gefundene Einträge<p>";
```

Seit der Version 4.0.5 ist es außerdem möglich parallele Suchen durchzuführen. Um dies zu verwirklichen benutzen Sie als erstes Argument einen Array von Verbindungs-Kennungen, statt einer einzelnen Verbindungs-Kennung. Falls Sie nicht den gleichen Basis DN und den gleichen Filter für alle Suchen verwenden möchten, können Sie ebenso einen Array von Basis DNs und/oder einen Array von Filtern benutzen. Diese Arrays müssen die gleiche Größe wie das Array der Verbindungs-Kennungen haben, da die ersten Einträge der Arrays für eine Suche verwendet werden,

die zweiten Einträge für eine andere Suche und so weiter. Wenn Sie parallel suchen erhalten Sie ein Array von Such-Ergebnis-Kennungen, außer im Fall eines Fehlers, dann liefert der Eintrag zur entsprechenden Suche `FALSE` zurück. Das entspricht ganz genau dem Wert der normalerweise zurückgeliefert wird, außer dass Sie immer eine Ergebnis-Kennung erhalten, wenn Sie eine Suche durchgeführt haben. Es treten einige seltene Fälle auf, wo eine normale Suche `FALSE` zurückgibt, während die parallele Suche eine Kennung zurückliefert.

ldap_set_option (PHP 4 >= 4.0.4)

Setzt den Wert der gegebenen Option

bool **ldap_set_option** (int Verbindungs-Kennung, int option, mixed neuerwert)

Setzt den Wert der bestimmten Option auf *`neuerwert`*.Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Der Parameter *`option`* kann einer der folgenden sein: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Hier eine kurze Beschreibung, schauen Sie unter draft-ietf-ldapext-ldap-c- api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt) nach
Details.

Die Optionen LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION und LDAP_OPT_ERROR_NUMBER haben einen ganzzahligen Wert, LDAP_OPT_REFERRALS und LDAP_OPT_RESTART haben einen boolschen Wert und die Optionen LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING und LDAP_OPT_MATCHED_DN bestehen aus einer Zeichenkette. Das erste Beispiel veranschaulicht deren Gebrauch. Die Optionen LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS benötigen eine Kontrolliste, d.h. der Wert muss ein Array aus Kontrollen sein. Eine Kontrolle besteht aus einer *oid* die die Kontrolle identifiziert, einem wahlweisen *wert*, und einem wahlweisen Kennzeichen für *criticality*. Bei PHP wird eine Kontrolle durch ein Array angegeben. Dieses Array enthält ein Element mit dem Schlüssel *oid* und einer Zeichenkette als Wert und zwei optionalen Elementen. Die optionalen Elemente sind Schlüssel *wert* mit einer Zeichenkette als Wert und dem Schlüssel *iscritical* mit einem boolschen Wert. Der vorgegebene Wert von *iscritical* ist *`FALSE`* falls Sie nichts angeben. Schauen Sie sich das zweite Beispiel weiter unten an.

> **Anmerkung:** Diese Funktion steht nur zur Verfügung, wenn Sie OpenLDAP 2.x.x ODER Netscape Directory SDK x.x verwenden und wurde in PHP 4.0.4 hinzugefügt.

Beispiel 1. Setzen der Protokollversion

```
// $ds ist eine gültige Verbindungs-Kennung für einen Verzeichnis-
// Server
```

```
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Verwenden von LDAPv3";
else
    echo "Kann das Protokoll nicht auf Version 3 setzen";
```

Beispiel 2. Setzen der Server Kontrollen

```
// $ds ist eine gültige Verbingund-Kennung für einen Verzeichnis-
// Server
// Kontrolle ohne Wert
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical hat den vorgegebenen Wert FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// Versuch, beide Kontrollen zu setzen
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1,
$ctrl2)))
    echo "Kann die Server Kontrollen nicht setzen";
```

Siehe auch ldap_get_option().

ldap_unbind (PHP 3, PHP 4 >= 4.0.0)

Unbind von einem LDAP Verzeichnis

int **ldap_unbind** (int Verbindungs-Kennung)

Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Die **ldap_unbind()** Funktion führt ein unbind auf ein LDAP Verzeichnis aus.

ldap_8859_to_t61 (PHP 4 >= 4.0.2)

Übersetzt 8859 Zeichen nach t61 Zeichen

string **ldap_8859_to_t61** (string value)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_first_reference (PHP 4 >= 4.0.5)

Liefert die erste Referenz

int **ldap_first_reference** (int link, int result)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_next_reference (PHP 4 >= 4.0.5)

Holt die nächste Referenz

int **ldap_next_reference** (int link, int entry)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_parse_reference (PHP 4 >= 4.0.5)

Gewinnt Informationen aus einem Referenz-Eintrag

boolean **ldap_parse_reference** (int link, int entry, array referrals)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_parse_result (PHP 4 >= 4.0.5)

Gewinnt Informationen aus einem Ergebnis

boolean **ldap_parse_result** (int link, int result, int errcode, string matcheddn, string errmsg, array referrals)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_set_rebind_proc (PHP 4 CVS only)

Set a callback function to do re-binds on referral chasing.

int **ldap_set_rebind_proc** (int link, string callback)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_sort (PHP 4 CVS only)

Sortiert LDAP Ergebniseinträge

int **ldap_sort** (int link, int result, string sortfilter)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_start_tls (PHP 4 CVS only)

Startet TLS

int **ldap_start_tls** (int link)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ldap_t61_to_8859 (PHP 4 >= 4.0.2)

Übersetzt t61 Zeichen nach 8859 Zeichen

string **ldap_t61_to_8859** (string value)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

XLIX. Mail Funktionen

Die mail() Funktionen erlauben das Versenden von Mail.

mail (PHP 3, PHP 4 >= 4.0.0)

Sende Mail

bool **mail** (string to, string subject, string message [, string additional_headers])

mail() veschickt die Nachricht in *message* an den Empfänger in *to*. Mehrere Empfänger können angegeben werden, indem ein Komma zwischen die Adressen in *to* gesetzt wird.

Beispiel 1. Verschicken von Mail.

```
mail("rasmus@lerdorf.on.ca", "My Subject", "Line 1\nLine 2\nLine 3");
```

Wenn ein vierter String-Parameter übergeben wird, dann wird dieser an das Ende des Mail-Headers gehängt. Mehrfache Header werden durch einen Zeilenumbruch getrennt.

Beispiel 2. Verschicke Mail mit zusätzlichen Headers.

```
mail("nobody@aol.com", "the subject", $message,
     "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-Mailer: PHP/" . p
```

L. mailparse functions

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

mailparse_uudecode_all (PHP 4 CVS only)

Scans the data from fp and extract each embedded uuencoded file. Returns an array listing filename information

array **mailparse_uudecode_all** (resource fp)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_rfc822_parse_addresses (PHP 4 >= 4.1.0)

Parse addresses and returns a hash containing that data

array **mailparse_rfc822_parse_addresses** (string addresses)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_determine_best_xfer_encoding (PHP 4 >= 4.1.0)

Figures out the best way of encoding the content read from the file pointer fp, which must be seek-able

int **mailparse_determine_best_xfer_encoding** (resource fp)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_stream_encode (PHP 4 >= 4.1.0)

Streams data from source file pointer, apply encoding and write to destfp

bool **mailparse_stream_encode** (resource sourcefp, resource destfp, string encoding)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_parse (PHP 4 >= 4.1.0)

Incrementally parse data into buffer

void **mailparse_msg_parse** (resource rfc2045buf, string data)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_parse_file (PHP 4 >= 4.1.0)

Parse file and return a resource representing the structure

resource **mailparse_msg_parse_file** (string filename)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_free (PHP 4 >= 4.1.0)

Frees a handle allocated by mailparse_msg_crea

void **mailparse_msg_free** (resource rfc2045buf)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_create (PHP 4 >= 4.1.0)

Returns a handle that can be used to parse a message

int **mailparse_msg_create** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_get_structure (PHP 4 >= 4.1.0)

Returns an array of mime section names in the supplied message

array **mailparse_msg_get_structure** (resource rfc2045)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_extract_part (PHP 4 >= 4.1.0)

Extracts/decodes a message section. If callbackfunc is not specified, the contents will be sent to "stdout"

void **mailparse_msg_extract_part** (resource rfc2045, string msgbody [, string callbackfunc])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_extract_part_file (PHP 4 >= 4.1.0)

Extracts/decodes a message section, decoding the transfer encoding

string **mailparse_msg_extract_part_file** (resource rfc2045, string filename [, string callbackfunc])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_get_part_data (PHP 4 >= 4.1.0)

Returns an associative array of info about the message

array **mailparse_msg_get_part_data** (resource rfc2045)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mailparse_msg_get_part (PHP 4 >= 4.1.0)

Returns a handle on a given section in a mimemessage

int **mailparse_msg_get_part** (resource rfc2045, string mimesection)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LI. Mathematische Funktionen

Einleitung

Die mathematischen Funktionen behandeln nur Werte im Bereich der Datentypen long und double der jeweiligen Architektur. Wenn Sie größere Zahlen oder Werte mit höherer Genauigkeit benötigen, sollten Sie sich die mathematische Funktionen mit beliebiger Genauigkeit ansehen.

Mathematische Konstanten

Die folgenden Werte werden von diesem Packet definiert:

Tabelle 1. Mathematische Konstanten

Konstante	Wert	Beschreibung
M_PI	3.14159265358979323846	Der Wert ¶ (Pi)
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)

Bis auf M_PI sind diese Konstanten erst ab PHP4.0 verfügbar.

abs (PHP 3, PHP 4 >= 4.0.0)

Absolutwert

mixed **abs** (mixed number)

Liefert den Absolutwert einer Zahl. Ist `number` vom Typ float, so ist auch das Ergebnis vom Typ float, ansonsten ist es vom Typ int.

acos (PHP 3, PHP 4 >= 4.0.0)

Arcuscosinus

float **acos** (float arg)

Liefert den Arcuscosinus zum Wert `arg` im Bogenmaß.

Siehe auch asin(), atan() und cos().

asin (PHP 3, PHP 4 >= 4.0.0)

Arcussinus

float **asin** (float arg)

Liefert den Arcussinus zum Wert `arg` im Bogenmaß.

Siehe auch acos(), atan() und sin().

atan (PHP 3, PHP 4 >= 4.0.0)

Arcustangens

float **atan** (float arg)

Liefert den Arcustangens zum Wert `arg` im Bogenmaß im Bereich (-¶/2 ; ¶/2].

See also asin(), acos(), atan2() und tan().

atan2 (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Arcustangens aus zwei Werten

float **atan2** (float y, float x)

Diese Funktion berechnet den Arcustangens aus den Parametern `x` und `y` im Bereich (-¶; ¶]. Dieser entspricht weitgehend dem Arcustangens aus x / y, allerdingskann hier das Vorzeichen beider Parameter ausgewertet und so der Quadrant des Ergebnises bestimmt werden.

Siehe auch atan(), acos(), asin(), und tan().

base_convert (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Konvertiert Werte zwischen Zahlensystemen unterschiedlicher Basis

strin **base_convert** (string number, int frombase, int tobase)

Diese Funktion liefert einen String, der `number` zur Basis `tobase` enthält. Der Parameter `frombase` giebt die Basis des Eingabewertes `number` an. `tobase` und `tobase` können Werte von 2 bis einschließlich 36 enthalten. Für die Darstellung von Werten mit einer Basis größer als 10 werden neben den Ziffern '0' bis '9' auch die Buchstaben von 'a' bis 'z' eingesetzt, wobei 'a' einem Wert von 10 entspricht, 'b' steht dementsprechend für den Wert 11, bis hin zum 'z', das den Wert 36 darstellt.

Beispiel 1. base_convert()

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec (PHP 3, PHP 4 >= 4.0.0)

Binär zu dezimal Konvertierung

int **bindec** (string binary_string)

Liefert die dezimale Entsprechnung der in `binary_string` übergebenen binären Zahl. Der größte konvertierbare Wert ist ein String aus 31 Einsen, das entspricht 2147483647 dezimal (2 hoch 32 minus 1).

Siehe auch decbin(), octdec(), und hexdec().

ceil (PHP 3, PHP 4 >= 4.0.0)

Aufrunden

int **ceil** (float number)

Diese Funktion liefert diejenige ganze Zahl, die größer oder gleich dem Parameter `number` ist.

Siehe auch floor() und round().

cos (PHP 3, PHP 4 >= 4.0.0)

Cosinus

float **cos** (float arg)

Liefert den Cosinus des Winkels `arg` (`arg` in Bogenmaß).

Siehe auch sin(), tan() und acos().

DecBin (PHP 3, PHP 4 >= 4.0.0)

Dezimal zu binär Konvertierung

string **decbin** (int number)

Diese Funktion liefert eine binäre Representation von `number` als String zurück. Der größte konvertierbare Wert für `number` ist dezimal 4294967295, dies entspricht (2 hoch 32)-1 und liefert einen String aus 32 Einsen.

Siehe auch bindec() sowie dechex() und decoct().

DecHex (PHP 3, PHP 4 >= 4.0.0)

Dezimal zu hexadezimal Konvertierung

string **dechex** (int number)

Diese Funktion liefert eine hexadezimale Representation von `number` als String zurück. Der größte konvertierbare Wert für `number` ist dezimal 2147483647, dies entspricht (2 hoch 32)-1 und liefet als Ergebnis den String "7fffffff" zurück.

Siehe auch hexdec() sowie decbin() und decoct().

DecOct (PHP 3, PHP 4 >= 4.0.0)

Dezimal zu oktal Konvertierung

string **decoct** (int number)

Diese Funktion liefert eine oktale (zur Basis 8) Representation von `number` als String zurück. Der größte konvertierbare Wert für `number` ist dezimal 2147483647, dies entspricht (2 hoch 32)-1 und liefet als Ergebnis den String "17777777777" zurück.

Siehe auch octdec() sowie decbin() und dechex().

deg2rad (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Grad in Bogenmaß wandeln

double **deg2rad** (double number)

Diese Funktion wandelt den übergebenen Winkel von Grad in Bogenmaß.

Siehe auch rad2deg().

exp (PHP 3, PHP 4 >= 4.0.0)

e hoch ...

float **exp** (float arg)

Liefert e potenziert mit *arg*.

Siehe auch pow().

floor (PHP 3, PHP 4 >= 4.0.0)

Abrunden

int **floor** (float number)

Diese Funktion liefert die ganze Zahl, die kleiner oder gleich dem Parameter *number* ist.

Siehe auch ceil() und round().

getrandmax (PHP 3, PHP 4 >= 4.0.0)

Größtmöglicher Zufallswert

int **getrandmax** (void)

Diese Funktion liefert den höchsten Wert, den ein Aufruf der Funktion rand() auf dieser Plattform zurückgeben kann.

Siehe auch rand(), srand() mt_rand(), mt_srand() und mt_getrandmax().

HexDec (PHP 3, PHP 4 >= 4.0.0)

Hexadezimal zu dezimal Konvertierung

int **hexdec** (string hex_string)

Liefert die dezimale Entsprechnung der in *hex_string* übergebenen hexadezimalen Zahl. Der größte konvertierbare Wert ist "7fffffff", das entspricht 2147483647 dezimal (2 hoch 32 minus 1).

Siehe auch dechex() sowie bindec() und octdec().

log (PHP 3, PHP 4 >= 4.0.0)

Natürlicher Logarithmus

float **log** (float arg)

Diese Funktion liefert den natürlichen Logarithmus (zur Basis e) des Parameters `arg`.

Siehe auch exp() und log10().

log10 (PHP 3, PHP 4 >= 4.0.0)

Zehner-Logarithmus

float **log10** (float arg)

Diese Funktion liefert den Logarithmus des Parameters `arg` zur Basis zehn.

Siehe auch log().

max (PHP 3, PHP 4 >= 4.0.0)

Maximalwert bestimmen

mixed **max** (mixed arg1, mixed arg2, mixed argn)

max() liefert den numerisch größten Eingabewert zurück.

Ist der erste Parameter ein Array, so wird der größte Wert innerhalb dieses Array zurückgegeben. Anderenfalls müßen mindestens zwei Parameter übergeben werden, und **max()** liefert den Wert des größten Parameters zurück. Die Anzahl der Eingabe-Paramter ist nur durch den verfügbaren Hauptspeicher begrenzt.

Ist mindestens einer der Eigabewerte vom Typ double, so werden alle Werte als double behandelt und der Rückgabewert ist ebenfalls vom Typ double. Anderenfalls werden alle Eingabewerte als int behandelt und auch ein int zurückgegeben.

Siehe auch min().

min (PHP 3, PHP 4 >= 4.0.0)

Minimalwert bestimmen

mixed **min** (mixed arg1, mixed arg2, mixed argn)

min() liefert den numerisch kleinsten Eingabewert zurück.

Ist der erste Parameter ein Array, so wird der kleinste Wert innerhalb dieses Array zurückgegeben. Anderenfalls müßen mindestens zwei Parameter übergeben werden, und **min()** liefert den Wert des kleinsten Parameters zurück. Die Anzahl der Eingabe-Paramter ist nur durch den verfügbaren Hauptspeicher begrenzt.

Ist mindestens einer der Eigabewerte vom Typ double, so werden alle Werte als double behandelt und der Rückgabewert ist ebenfalls vom Typ double. Anderenfalls werden alle Eingabewerte als int behandelt und auch ein int zurückgegeben.

Siehe auch max().

mt_rand (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Erzeugt ’bessere’ Zufallszahlen

int **mt_rand** ([int min [, int max]])

Die PHP rand() Funktion benutzt den in der jeweiligen libc enthaltenen Zufallsgenerator. Gerade in älteren libc-Versionen ist dieser aber oft langsam und die Qualität der erzeugten Pseudozufalls-Sequenzen ist unbefriedigend bis unbekannt.

Die Funktion **mt_rand()** bietet einen Drop-In Ersatz für die von der libc bereitgestellten Funktionen. Sie benutzt einen Zufallsgenerator mit bekannter Characteristik, den sogenannten "Mersenne Twister", der Pseudozufalls-Sequenzen generiert, die auch kryptografischen Ansprüchen genügen sollten und viermal so schnell arbeitet wie der typische libc-Algorithmus.

Weitere Informationen zum Mersenne Twister und zur opmimierten Implementation finden sich unter http://www.math.keio.ac.jp/~matumoto/emt.html und http://www.scp.syr.edu/~marc/hawk/twister.html.

Ohne die optionalen Parameter *`min`* und *`max`* liefert **mt_rand()** eine Pseudo-Zufallszahl zwischen 0 und einem plattformabhängigen Maximalwert, der mit mt_getrandmax() ermittelt werden kann.

Benutzen Sie unbedingt mt_srand(), um vor dem ersten Aufruf von **mt_rand()** einen neuen Startwert festzulegen, ansonsten erhalten Sie bei jedem Programmlauf dieselbe Sequenz von Werten.

> **Anmerkung:** In PHP-Versionen vor 3.0.7 hatte der zweite Parameter *`max`* nicht den Maximalwert sondern die Größe des Werteintervalls an, d.h. in diesen älteren Version muß statt `mt_rand(min,max)` `mt_rand(min,max-min)` benutzt werden, um dasselbe Ergebnis zu erhalten.

Siehe auch mt_srand(), mt_getrandmax(), srand(), rand() und getrandmax().

mt_srand (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

seed the better random number generator

void **mt_srand** (int seed)

Legt einen internen Startwert für den Zufallsgenerator in mt_rand() fest. Gleiche Startwerte führen auch jeweils zu derselben Pseudo-Zufallssequenzen daher ist es üblich, diese über die aktuelle Uhrzeit festzulegen:

```
// seed with microseconds since last "whole" second
mt_srand((double)microtime()*1000000);
$randval = mt_rand();
```

Siehe auch mt_rand(), mt_getrandmax(), srand(), rand() und getrandmax().

mt_getrandmax (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

show largest possible random value

int **mt_getrandmax** (void)

Diese Funktion liefert den höchsten Wert, den ein Aufruf der Funktion mt_rand() auf dieser Plattform zurückgeben kann.

Siehe auch mt_rand(), mt_srand() rand(), srand() and getrandmax().

number_format (PHP 3, PHP 4 >= 4.0.0)

Formartiert eine Zahl mit Tausendergruppen

string **number_format** (float number [, int decimals [, string dec_point [, string thousands_sep]]])

number_format() erzeugt aus *number* einen formatierten String. Die Funktion akzeptiert ein, zwei oder vier Parameter (nicht drei).

Wird nur *number* übergeben, so wird diese Zahl ohne Nachkommastellen, aber mit durch Komma getrennten Tausendergruppen ausgegeben. (1234567 -> 1,234,567)

Der optionale Parameter *decimals* giebt die Anzahl der auszugebenden Nachkommastellen an, diese werden durch einen Punkt getrennt ausgegeben.

Mit Hilfe der weiteren Parameter *dec_point* und *thousands_sep* können andere Trennzeichen für Tausendergruppen und Nachkommastellen festgelegt werden. Diese beiden Parameter können nur zusammen angegeben werden.

Im deutschen Sprachraum werden die Trennzeichen i.A. genau umgekehrt benutzt, d.h. Nachkommastellen werden durch ein Komma und Tausender durch einen Punkt getrennt.

OctDec (PHP 3, PHP 4 >= 4.0.0)

Oktal zu dezimal Konvertierung

int **octdec** (string octal_string)

Liefert die dezimale Entsprechnung der in *octal_string* übergebenen oktalen Zahl. Der größte konvertierbare Wert ist "17777777777", das entspricht 2147483647 dezimal (2 hoch 32 minus 1).

Siehe auch decoct() sowie bindec() und hexdec().

pi (PHP 3, PHP 4 >= 4.0.0)

Wert von PI

double **pi** (void)

Diese Funktionen liefert (im Rahmen der möglichen Genauigkeit) den Wert der Kreiszahl Pi. Die Konstante M_PI erfüllt denselben Zweck allerdings wesentlich effektiver.

pow (PHP 3, PHP 4 >= 4.0.0)

Exponentialfunktion

float **pow** (float base, float exp)

Diese Funktion liefert `base` potenziert mit `exp`, also `base` hoch `exp`.

Siehe auch exp().

rad2deg (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Bogenmaß in Grad wandeln

double **rad2deg** (double number)

Diese Funktion wandelt den übergebenen Winkel von Bogenmaß in Grad.

Siehe auch deg2rad().

rand (PHP 3, PHP 4 >= 4.0.0)

(Pseudo-) Zufallszahl generieren

int **rand** ([int min [, int max]])

Diese Funktion liefert eine Pseudo-Zufallszahl zwischen `min` und `max` (inclusive). Ohne die optionalen Parameter `min` und `max` liefert **rand()** eine Pseudo-Zufallszahl zwischen 0 und einem plattformabhängigen Maximalwert, der mit getrandmax() ermittelt werden kann.

Benutzen Sie unbedingt srand(), um vor dem ersten Aufruf von **rand()** einen neuen Startwert festzulegen, ansonsten erhalten Sie bei jedem Programmlauf dieselbe Sequenz von Werten.

> **Anmerkung:** In PHP-Versionen vor 3.0.7 hatte der zweite Parameter `max` nicht den Maximalwert sondern die Größe des Werteintervalls an, d.h. in diesen älteren Version muß statt `rand (min, max) rand (min, max-min)` benutzt werden, um dasselbe Ergebnis zu erhalten.

Siehe auch srand(), getrandmax(), mt_rand(), mt_srand() und mt_getrandmax().

round (PHP 3, PHP 4 >= 4.0.0)

Runden eines Wertes

double **round** (double val [, int precision])

Diese Funktion rundet den Eingabewert *val* auf *precision* Stellen nach dem Komma auf oder ab.

```
$foo = round (3.4);   // $foo == 3.0
$foo = round (3.5);   // $foo == 4.0
$foo = round (3.6);   // $foo == 4.0

$foo = round (1.95583,2);   // $foo == 1.96
```

Anmerkung: Der Parameter *precision* ist nur in PHP 4 verfügbar.

Siehe auch ceil() und floor().

sin (PHP 3, PHP 4 >= 4.0.0)

Sinus

float **sin** (float arg)

Liefert den Sinus des Eingabewertes *arg*. *arg* wird im Bogenmaß angegeben.

Siehe auch cos(), tan() und asin().

sqrt (PHP 3, PHP 4 >= 4.0.0)

Quadratwurzel

float **sqrt** (float arg)

Diese Funktion liefert die Quadratwurzel des Wertes *arg*.

srand (PHP 3, PHP 4 >= 4.0.0)

Startwert für den Zufallsgenerator festlegen

void **srand** (int seed)

Legt einen internen Startwert für den Zufallsgenerator in rand() fest. Gleiche Startwerte führen auch jeweils zu derselben Pseudo-Zufallssequenzen daher ist es üblich, diese über die aktuelle Uhrzeit festzulegen:

```
// seed with microseconds since last "whole" second
srand ((double)microtime()*1000000);
$randval = rand();
```

Siehe auch rand(), getrandmax(), mt_rand(), mt_srand() und mt_getrandmax().

tan (PHP 3, PHP 4 >= 4.0.0)

Tangens

float **tan** (float arg)

Liefert den Tangens des Wertes `arg`. `arg` wird dabei im Bogenmaß angegeben.

Siehe auch sin(), cos(), atan() und atan2().

LII. Multi-Byte String Functions

Introduction

There are many languages in which all characters can be expressed by single byte. Multi-byte character codes are used to express many characters for many languages. `mbstring` is developed to handle Japanese characters. However, many `mbstring` functions are able to handle character encoding other than Japanese.

A multi-byte character encoding represents single character with consecutive bytes. Some character encoding has shift(escape) sequences to start/end multi-byte character strings. Therefore, a multi-byte character string may be destroyed when it is divided and/or counted unless multi-byte character encoding safe method is used. This module provides multi-byte character safe string functions and other utility functions such as conversion functions.

Since PHP is basically designed for ISO-8859-1, some multi-byte character encoding does not work well with PHP. Therefore, it is important to set `mbstring.internal_encoding` to a character encoding that works with PHP.

PHP4 Character Encoding Requirements

- Per byte encoding
- Single byte characters in range of `00h-7fh` which is compatible with `ASCII`
- Multi-byte characters without `00h-7fh`

These are examples of internal character encoding that works with PHP and does NOT work with PHP.

```
Character encodings work with PHP:
ISO-8859-*, EUC-JP, UTF-8
```

```
Character encodings do NOT work with PHP:
JIS, SJIS
```

Character encoding, that does not work with PHP, may be converted with `mbstring`'s HTTP input/output conversion feature/function.

> **Anmerkung:** SJIS should not be used for internal encoding unless the reader is familiar with parser/compiler, character encoding and character encoding issues.

> **Anmerkung:** If you use database with PHP, it is recommended that you use the same character encoding for both database and `internal encoding` for ease of use and better performance.
>
> If you are using PostgreSQL, it supports character encoding that is different from backend

character encoding. See the PostgreSQL manual for details.

How to Enable mbstring

`mbstring` is an extended module. You must enable module with `configure` script. Refer to the Install section for details.

The following configure options are related to `mbstring` module.

- `--enable-mbstring` : Enable `mbstring` functions. This option is required to use `mbstring` functions.
- `--enable-mbstr-enc-trans` : Enable HTTP input character encoding conversion using `mbstring` conversion engine. If this feature is enabled, HTTP input character encoding may be converted to `mbstring.internal_encoding` automatically.

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

If `enctype` for HTML form is set to `multipart/form-data`, `mbstring` does not convert character encoding in POST data. If it is the case, strings are needed to be converted to internal character encoding.

- HTTP Input

  There is no way to control HTTP input character conversion from PHP script. To disable HTTP input character conversion, it has to be done in `php.ini`.

  Beispiel 1. Disable HTTP input conversion in php.ini

  ```
  ;; Disable HTTP Input conversion
  mbstring.http_input = pass
  ```

  When using PHP as an Apache module, it is possible to override PHP ini setting per Virtual Host in `httpd.conf` or per directory with `.htaccess`. Refer to the Configuration section and Apache Manual for details.

- HTTP Output

  There are several ways to enable output character encoding conversion. One is using `php.ini`, another is using ob_start() with mb_output_handler() as `ob_start` callback function.

  Anmerkung: For PHP3-i18n users, `mbstring`'s output conversion differs from PHP3-i18n.

Character encoding is converted using output buffer.

Beispiel 2. `php.ini` setting example

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
output_handler      = mb_output_handler
```

Beispiel 3. Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encoding

Currently, the following character encoding is supported by `mbstring` module. Caracter encoding may be specified for `mbstring` functions' `encoding` parameter.

The following character encoding is supported in this PHP extension :

`UCS-4`, `UCS-4BE`, `UCS-4LE`, `UCS-2`, `UCS-2BE`, `UCS-2LE`, `UTF-32`, `UTF-32BE`, `UTF-32LE`, `UCS-2LE`, `UTF-16`, `UTF-16BE`, `UTF-16LE`, `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`, `SJIS`, `eucJP-win`, `SJIS-win`, `ISO-2022-JP`, `JIS`, `ISO-8859-1`, `ISO-8859-2`, `ISO-8859-3`, `ISO-8859-4`, `ISO-8859-5`, `ISO-8859-6`, `ISO-8859-7`, `ISO-8859-8`, `ISO-8859-9`, `ISO-8859-10`,

ISO-8859-13, ISO-8859-14, ISO-8859-15, byte2be, byte2le, byte4be, byte4le, BASE64, 7bit, 8bit and UTF7-IMAP.

php.ini entry, which accepts encoding name, accepts "auto" and "pass" also. mbstring functions, which accepts encoding name, and accepts "auto".

If "pass" is set, no character encoding conversion is performed.

If "auto" is set, it is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS".

See also mb_detect_order()

> **Anmerkung:** "Supported character encoding" does not mean that it works as internal character code.

php.ini settings

- mbstring.internal_encoding defines default internal character encoding.
- mbstring.http_input defines default HTTP input character encoding.
- mbstring.http_output defines default HTTP output character encoding.
- mbstring.detect_order defines default character code detection order. See also mb_detect_order().
- mbstring.substitute_character defines character to substitute for invalid character encoding.

Web Browsers are supposed to use the same character encoding when submitting form. However, browsers may not use the same character encoding. See mb_http_input() to detect character encoding used by browsers.

If enctype is set to multipart/form-data in HTML forms, mbstring does not convert character encoding in POST data. The user must convert them in the script, if conversion is needed.

Although, browsers are smart enough to detect character encoding in HTML. charset is better to be set in HTTP header. Change default_charset according to character encoding.

Beispiel 4. php.ini setting example

```
;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to "ASCII,JIS,UTF-8,EUC-
JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order
```

```
;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Beispiel 5. `php.ini` setting for `EUC-JP` users

```
;; Disable Output Buffering
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto

;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Beispiel 6. `php.ini` setting for `SJIS` users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set http input encoding conversion to auto
```

```
mbstring.http_input  = auto

;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Overload of PHP string functions by mbstring functions with multibyte support

Because almost PHP application written for language using single-byte character encoding, there are some difficulties for multibyte string handling including japanese. Almost PHP string functions such as substr() do not support multibyte string.

Multibyte extension (mbstring) has some PHP string functions with multibyte support (ex. substr() supports mb_substr()).

Multibyte extension (mbstring) also supports ’function overloading’ to add multibyte string functionality without code modification. Using function overloading, some PHP string functions will be oveloaded multibyte string functions. For example, mb_substr() is called instead of substr() if function overloading is enabled. Function overload makes easy to port application supporting only single-byte encoding for multibyte application.

`mbstring.func_overload` in php.ini should be set some positive value to use function overloading. The value should specify the category of overloading functions, sbould be set 1 to enable mail function overloading. 2 to enable string functions, 4 to regular expression functions. For example, if is set for 7, mail, strings, regex functions should be overloaded. The list of overloaded functions are shown in below.

Tabelle 1. Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	mail()	mb_send_mail()
2	strlen()	mb_strlen()
2	strpos()	mb_strpos()
2	strrpos()	mb_strrpos()
2	substr()	mb_substr()
4	ereg()	mb_ereg()
4	eregi()	mb_eregi()
4	ereg_replace()	mb_ereg_replace()
4	eregi_replace()	mb_eregi_replace()
4	split()	mb_split()

Basics for Japanese multi-byte character

Most Japanese characters need more than 1 byte per character. In addition, several character encoding schemas are used under a Japanese environment. There are EUC-JP, Shift_JIS(SJIS) and ISO-2022-JP(JIS) character encoding. As Unicode becomes popular, UTF-8 is used also. To develop Web applications for a Japanese environment, it is important to use the character set for the task in hand, whether HTTP input/output, RDBMS and E-mail.

- Storage for a character can be up to six bytes
- A multi-byte character is usually twice of the width compared to single-byte characters. Wider characters are called "zen-kaku" - meaning full width, narrower characters are called "han-kaku" - meaning half width. "zen-kaku" characters are usually fixed width.
- Some character encoding defines shift(escape) sequence for entering/exiting multi-byte character strings.
- ISO-2022-JP must be used for SMTP/NNTP.
- "i-mode" web site is supposed to use SJIS.

References

Multi-byte character encoding and its related issues are very complex. It is impossible to cover in sufficient detail here. Please refer to the following URLs and other resources for further readings.

- Unicode/UTF/UCS/etc

  `http://www.unicode.org/`

- Japanese/Korean/Chinese character information

  `ftp://ftp.ora.com/pub/examples/nutshell/ujip/doc/cjk.inf`

mb_language (PHP 4 >= 4.0.6)

Set/Get current language

string **mb_language** ([string language])

mb_language() sets language. If `language` is omitted, it returns current language as string.

`language` setting is used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). mb_send_mail() uses this setting to encode e-mail.

Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Value: If `language` is set and `language` is valid, it returns `TRUE`. Otherwise, it returns `FALSE`. When `language` is omitted, it returns language name as string. If no language is set previously, it returns `FALSE`.

See also mb_send_mail().

mb_parse_str (PHP 4 >= 4.0.6)

Parse GET/POST/COOKIE data and set global variable

boolean **mb_parse_str** (string encoded_string [, array result])

mb_parse_str() parses GET/POST/COOKIE data and sets global variables. Since PHP does not provide raw POST/COOKIE data, it can only used for GET data for now. It preses URL encoded data, detects encoding, converts coding to internal encoding and set values to `result` array or global variables.

`encoded_string`: URL encoded data.

`result`: Array contains decoded and character encoding converted values.

Return Value: It returns `TRUE` for success or `FALSE` for failure.

See also mb_detect_order(), mb_internal_encoding().

mb_internal_encoding (PHP 4 >= 4.0.6)

Set/Get internal character encoding

string **mb_internal_encoding** ([string encoding])

mb_internal_encoding() sets internal character encoding to `encoding` If parameter is omitted, it returns current internal encoding.

`encoding` is used for HTTP input character encoding conversion, HTTP output character encoding conversion and default character encoding for string functions defined by mbstring module.

`encoding`: Character encoding name

Return Value: If `encoding` is set,**mb_internal_encoding()** returns `TRUE` for success, otherwise returns `FALSE`. If `encoding` is omitted, it returns current character encoding name.

Beispiel 1. mb_internal_encoding() example

```
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");

/* Display current internal character encoding */
echo mb_internal_encoding();
```

See also mb_http_input(), mb_http_output(), mb_detect_order().

mb_http_input (PHP 4 >= 4.0.6)

Detect HTTP input character encoding

string **mb_http_input** ([string type])

mb_http_input() returns result of HTTP input character encoding detection.

type: Input string specifies input type. "G" for GET, "P" for POST, "C" for COOKIE. If type is omitted, it returns last input type processed.

Return Value: Character encoding name. If **mb_http_input()** does not process specified HTTP input, it returns `FALSE`.

See also mb_internal_encoding(), mb_http_output(), mb_detect_order().

mb_http_output (PHP 4 >= 4.0.6)

Set/Get HTTP output character encoding

string **mb_http_output** ([string encoding])

If *encoding* is set, **mb_http_output()** sets HTTP output character encoding to *encoding*. Output after this function is converted to *encoding*. **mb_http_output()** returns `TRUE` for success and `FALSE` for failure.

If *encoding* is omitted, **mb_http_output()** returns current HTTP output character encoding.

See also mb_internal_encoding(), mb_http_input(), mb_detect_order().

mb_detect_order (PHP 4 >= 4.0.6)

Set/Get character encoding detection order

array **mb_detect_order** ([mixed encoding-list])

mb_detect_order() sets automatic character encoding detection order to *`encoding-list`*. It returns `TRUE` for success, `FALSE` for failure.

`encoding-list` is array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS")

If *`encoding-list`* is omitted, it returns current character encoding detection order as array.

This setting affects mb_detect_encoding() and mb_send_mail().

> **Anmerkung:** `mbstring` currently implements following encoding detection filters. If there is a invalid byte sequence for following encoding, encoding detection will fail.
>
> **Anmerkung:** `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`,`SJIS`, `eucJP-win`, `SJIS-win`, `JIS`, `ISO-2022-JP`
>
> For `ISO-8859-*`, `mbstring` always detects as `ISO-8859-*`.
>
> For `UTF-16`, `UTF-32`, `UCS2` and `UCS4`, encoding detection will fail always.

Beispiel 1. Useless detect order example

```
; Always detect as ISO-8859-1
detect_order = ISO-8859-1, UTF-8

; Always detect as UTF-8, since ASCII/UTF-7 values are
; valid for UTF-8
detect_order = UTF-8, ASCII, UTF-7
```

Beispiel 2. mb_detect_order() examples

```
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
```

See also mb_internal_encoding(), mb_http_input(), mb_http_output() mb_send_mail().

mb_substitute_character (PHP 4 >= 4.0.6)

Set/Get substitution character

mixed **mb_substitute_character** ([mixed substrchar])

mb_substitute_character() specifies substitution character when input character encoding is invalid or character code is not exist in output character encoding. Invalid characters may be substituted `NULL`(no output), string or integer value (Unicode character code value).

This setting affects mb_detect_encoding() and mb_send_mail().

`substchar` : Specify Unicode value as integer or specify as string as follows

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Value: If *`substchar`* is set, it returns `TRUE` for success, otherwise returns `FALSE`. If *`substchar`* is not set, it returns Unicode value or "`none`"/"`long`".

Beispiel 1. mb_substitute_character() example

```
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
```

mb_output_handler (PHP 4 >= 4.0.6)

Callback function converts character encoding in output buffer

string **mb_output_handler** (string contents, int status)

mb_output_handler() is ob_start() callback function. **mb_output_handler()** converts characters in output buffer from internal character encoding to HTTP output character encoding.

4.1.0 or later version, this hanlder adds charset HTTP header when following conditions are met:

- Does not set `Content-Type` by header()
- Default MIME type begins with `text/`
- `http_output` setting is other than pass

contents : Output buffer contents

status : Output buffer status

Return Value: String converted

Beispiel 1. mb_output_handler() example

```
mb_http_output("UTF-8");
ob_start("mb_output_handler");
```

Anmerkung: If you want to output some binary data such as image from PHP script, you must set output encoding to "pass" using mb_http_output().

See also ob_start().

mb_preferred_mime_name (PHP 4 >= 4.0.6)

Get MIME charset string

string **mb_preferred_mime_name** (string encoding)

mb_preferred_mime_name() returns MIME `charset` string for character encoding *`encoding`*. It returns `charset` string.

Beispiel 1. mb_preferred_mime_string() example

```
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
```

mb_strlen (PHP 4 >= 4.0.6)

Get string length

string **mb_strlen** (string str [, string encoding])

mb_strlen() returns number of characters in string *str* having character encoding *encoding*. A multi-byte character is counted as 1.

encoding is character encoding for *str*. If *encoding* is omitted, internal character encoding is used.

See also mb_internal_encoding(), strlen().

mb_strpos (PHP 4 >= 4.0.6)

Find position of first occurrence of string in a string

int **mb_strpos** (string haystack, string needle [, int offset [, string encoding]])

mb_strpos() returns the numeric position of the first occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns FALSE.

mb_strpos() performs multi-byte safe strpos() operation based on number of characters. *needle* position is counted from the beginning of the *haystack*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal character encoding is used. mb_strrpos() accepts string for *needle* where strrpos() accepts only character.

offset is search offset. If it is not specified, 0 is used.

encoding is character encoding name. If it is omitted, internal character encoding is used.

See also **mb_strpos()**, mb_internal_encoding(), strpos()

mb_strrpos (PHP 4 >= 4.0.6)

Find position of last occurrence of a string in a string

int **mb_strrpos** (string haystack, string needle [, string encoding])

mb_strrpos() returns the numeric position of the last occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns FALSE.

mb_strrpos() performs multi-byte safe strrpos() operation based on number of characters. *needle* position is counted from the beginning of *haystack*. First character's position is 0. Second character position is 1.

If *encoding* is omitted, internal encoding is assumed. **mb_strrpos()** accepts string for *needle* where strrpos() accepts only character.

encoding is character encoding. If it is not specified, internal character encoding is used.

See also mb_strpos(), mb_internal_encoding(), strrpos().

mb_substr (PHP 4 >= 4.0.6)

Get part of string

string **mb_substr** (string str, int start [, int length [, string encoding]])

mb_substr() returns the portion of *str* specified by the *start* and *length* parameters.

mb_substr() performs multi-byte safe substr() operation based on number of characters. Position is counted from the beginning of *str*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal encoding is assumed.

encoding is character encoding. If it is omitted, internal character encoding is used.

See also mb_strcut(), mb_internal_encoding().

mb_strcut (PHP 4 >= 4.0.6)

Get part of string

string **mb_strcut** (string str, int start [, int length [, string encoding]])

mb_strcut() returns the portion of *str* specified by the *start* and *length* parameters.

mb_strcut() performs equivalent operation as mb_substr() with different method. If *start* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *str* that is shorter than *length* AND character that is not part of multi-byte string or not being middle of shift sequence.

encoding is character encoding. If it is not set, internal character encoding is used.

See also mb_substr(), mb_internal_encoding().

mb_strwidth (PHP 4 >= 4.0.6)

Return width of string

int **mb_strwidth** (string str [, string encoding])

mb_strwidth() returns width of string *str*.

Multi-byte character usually twice of width compare to single byte character.

```
Character width

U+0000 - U+0019   0
U+0020 - U+1FFF   1
U+2000 - U+FF60   2
U+FF61 - U+FF9F   1
U+FFA0 -          2
```

encoding is character encoding. If it is omitted, internal encoding is used.

See also: mb_strimwidth(), mb_internal_encoding().

mb_strimwidth (PHP 4 >= 4.0.6)

Get truncated string with specified width

string **mb_strimwidth** (string str, int start, int width, string trimmarker [, string encoding])

mb_strimwidth() truncates string *str* to specified *width*. It returns truncated string.

If *trimmarker* is set, *trimmarker* is appended to return value.

start is start position offset. Number of characters from the beginning of string. (First character is 0)

trimmarker is string that is added to the end of string when string is truncated.

encoding is character encoding. If it is omitted, internal encoding is used.

Beispiel 1. mb_strimwidth() example

```
$str = mb_strimwidth($str, 0, 40, "..>");
```

See also: mb_strwidth(), mb_internal_encoding().

mb_convert_encoding (PHP 4 >= 4.0.6)

Convert character encoding

string **mb_convert_encoding** (string str, string to-encoding [, mixed from-encoding])

mb_convert_encoding() converts character encoding of string *str* from *from-encoding* to *to-encoding*.

str : String to be converted.

from-encoding is specified by character code name before conversion. it can be array or string - comma separated enumerated list.

Beispiel 1. mb_convert_encoding() example

```
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");
```

```
/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-
2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
```

See also: mb_detect_order().

mb_detect_encoding (PHP 4 >= 4.0.6)

Detect character encoding

string **mb_detect_encoding** (string str [, mixed encoding-list])

mb_detect_encoding() detects character encoding in string *str*. It returns detected character encoding.

encoding-list is list of character encoding. Encoding order may be specified by array or comma separated list string.

If *encoding_list* is omitted, detect_order is used.

Beispiel 1. mb_detect_encoding() example

```
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");

/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
```

See also: mb_detect_order().

mb_convert_kana (PHP 4 >= 4.0.6)

Convert "kana" one from another ("zen-kaku" ,"han-kaku" and more)

string **mb_convert_kana** (string str, string option [, mixed encoding])

mb_convert_kana() performs "han-kaku" - "zen-kaku" conversion for string *str*. It returns converted string. This function is only useful for Japanese.

option is conversion option. Default value is "KV".

encoding is character encoding. If it is omitted, internal character encoding is used.

```
Applicable Conversion Options

option : Specify with conversion of following options. Default "KV"
"r" :  Convert "zen-kaku" alphabets to "han-kaku"
"R" :  Convert "han-kaku" alphabets to "zen-kaku"
"n" :  Convert "zen-kaku" numbers to "han-kaku"
"N" :  Convert "han-kaku" numbers to "zen-kaku"
"a" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
"A" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
(Characters included in "a", "A" options are
U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
"s" :  Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
"S" :  Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
"k" :  Convert "zen-kaku kata-kana" to "han-kaku kata-kana"
"K" :  Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
"h" :  Convert "zen-kaku hira-gana" to "han-kaku kata-kana"
"H" :  Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
"c" :  Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
"C" :  Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
"V" :  Collapse voiced sound notation and convert them into a charac-
ter. Use with "K","H"
```

Beispiel 1. mb_convert_kana() example

```
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
   and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
```

mb_encode_mimeheader (PHP 4 >= 4.0.6)

Encode string for MIME header

string **mb_encode_mimeheader** (string str [, string charset [, string transfer-encoding [, string linefeed]]])

mb_encode_mimeheader() converts string *str* to encoded-word for header field. It returns converted string in ASCII encoding.

charset is character encoding name. Default is `ISO-2022-JP`.

transfer-encoding is transfer encoding. It should be one of `"B"` (Base64) or `"Q"` (Quoted-Printable). Default is `"B"`.

linefeed is end of line marker. Default is `"\r\n"` (CRLF).

Beispiel 1. mb_convert_kana() example

```
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $do-
ma . ">";
echo $addr;
```

See also mb_decode_mimeheader().

mb_decode_mimeheader (PHP 4 >= 4.0.6)

Decode string in MIME header field

string **mb_decode_mimeheader** (string str)

mb_decode_mimeheader() decodes encoded-word string *str* in MIME header.

It returns decoded string in internal character encoding.

See also mb_encode_mimeheader().

mb_convert_variables (PHP 4 >= 4.0.6)

Convert character code in variable(s)

string **mb_convert_variables** (string to-encoding, mixed from-encoding, mixed vars)

mb_convert_variables() convert character encoding of variables *vars* in encoding *from-encoding* to encoding *to-encoding*. It returns character encoding before conversion for success, `FALSE` for failure.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

It *from-encoding* is specified by array or comma separated string, it tries to detect encoding from *from-coding*. When *encoding* is omitted, `detect_order` is used.

vars (3rd and larger) is reference to variable to be converted. String, Array and Object are accepted. **mb_convert_variables()** assumes all parameters have the same encoding.

Beispiel 1. mb_convert_variables() example

```
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
```

mb_encode_numericentity (PHP 4 >= 4.0.6)

Encode character to HTML numeric string reference

string **mb_encode_numericentity** (string str, array convmap [, string encoding])

mb_encode_numericentity() converts specified character codes in string *str* from HTML numeric character reference to character code. It returns converted string.

array is array specifies code area to convert.

encoding is character encoding.

Beispiel 1. *convmap* example

```
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Beispiel 2. mb_encode_numericentity() example

```
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
```

```
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
   string reference */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
```

See also: mb_decode_numericentity().

mb_decode_numericentity (PHP 4 >= 4.0.6)

Decode HTML numeric string reference to character

string **mb_decode_numericentity** (string str, array convmap [, string encoding])

Convert numeric string reference of string *str* in specified block to character. It returns converted string.

array is array to specifies code area to convert.

encoding is character encoding. If it is omitted, internal character encoding is used.

Beispiel 1. *convmap* example

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See also: mb_encode_numericentity().

mb_send_mail (PHP 4 >= 4.0.6)

Send encoded mail.

boolean **mb_send_mail** (string to, string subject, string message [, string additional_headers [, string additional_parameter]])

mb_send_mail() sends email. Headers and message are converted and encoded according to mb_language() setting. **mb_send_mail()** is wrapper function of mail(). See mail() for details.

to is mail addresses send to. Multiple recipients can be specified by putting a comma between each address in to. This parameter is not automatically encoded.

subject is subject of mail.

message is mail message.

additional_headers is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

additional_parameter is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Gibt bei Erfolg TRUE zurück, im Fehlerfall FALSE.

See also mail(), mb_encode_mimeheader(), and mb_language().

mb_get_info (PHP 4 CVS only)

Get internal settings of mbstring

string **mb_get_info** ([string type])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_get_info() returns internal setting parameter of mbstring.

If *type* isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload" will be returned.

If *type* is specified for "http_output", "http_input", "internal_encoding", "func_overload", the specified setting parameter will be returned.

See also mb_internal_encoding(), mb_http_output().

mb_regex_encoding (PHP 4 CVS only)

Returns current encoding for multibyte regex as string

string **mb_regex_encoding** ([string encoding])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_regex_encoding() returns the character encoding used by multibyte regex functions.

If the optional parameter `encoding` is specified, it is set to the character encoding for multibyte regex. The default value is the internal character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_internal_encoding(), mb_ereg()

mb_ereg (PHP 4 CVS only)

Regular expression match with multibyte support

int **mb_ereg** (string pattern, string string [, array regs])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg() executes the regular expression match with multibyte support, and returns 1 if matches are found. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray `regs` will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, `FALSE` will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi()

mb_eregi (PHP 4 CVS only)

Regular expression match ignoring case with multibyte support

int **mb_eregi** (string pattern, string string [, array regs])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_eregi() executes the regular expression match with multibyte support, and returns 1 if matches are found. This function ignore case. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray `regs` will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, `FALSE` will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_ereg_replace (PHP 4 CVS only)

Replace regular expression with multibyte support

string **mb_ereg_replace** (string pattern, string replacement, string string [, array option])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_replace() scans `string` for matches to `pattern`, then replaces the matched text with `replacement` and returns the result string or `FALSE` on error. Multibyte character can be used in `pattern`.

Matching condition can be set by `option` parameter. If `i` is specified for this parameter, the case will be ignored. If `x` is specified, white space will be ignored. If `m` is specified, match will be executed in multiline mode and line break will be included in '.'. If `p` is specified, match will be executed in POSIX mode, line break will be considered as normal character. If `e` is specified, `replacement` string will be evaluated as PHP expression.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi_replace().

mb_eregi_replace (PHP 4 CVS only)

Replace regular expression with multibyte support ignoring case

string **mb_eregi_replace** (string pattern, string replace, string string)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_replace() scans `string` for matches to `pattern`, then replaces the matched text with `replacement` and returns the result string or `FALSE` on error. Multibyte character can be used in `pattern`. The case will be ignored.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_replace().

mb_split (PHP 4 CVS only)

Split multibyte string using regular expression

array **mb_split** (string pattern, string string [, int limit])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_split() split multibyte `string` using regular expression `pattern` and returns the result as an array.

If optional parameter `limit` is specified, it will be split in `limit` elements as maximum.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_ereg_match (PHP 4 CVS only)

Regular expression match for multibyte string

bool **mb_ereg_match** (string pattern, string string [, string option])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_match() returns TRUE if *string* matches regular expression *pattern*, FALSE if not.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_ereg_search (PHP 4 CVS only)

Multibyte regular expression match for predefined multibyte string

bool **mb_ereg_search** ([string pattern [, string option]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search() returns TRUE if the multibyte string matches with the regular expression, FALSE for otherwise. The string for matching is set by mb_ereg_search_init(). If *pattern* is not specified, the previous one is used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_pos (PHP 4 CVS only)

Return position and length of matched part of multibyte regular expression for predefined multibyte string

array **mb_ereg_search_pos** ([string pattern [, string option]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search_pos() returns an array including position of matched part for multibyte regular expression. The first element of the array will be the beggining of matched part, the second element will be length (bytes) of matched part. It returns FALSE on error.

The string for match is specified by mb_ereg_search_init(). It it is not specified, the previous one will be used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_regs (PHP 4 CVS only)

Returns the matched part of multibyte regular expression

array **mb_ereg_search_regs** ([string pattern [, string option]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first grouped part with brackets as second element, the second grouped part as third element, and so on. It returns FALSE on error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_init (PHP 4 CVS only)

Setup string and regular expression for multibyte regular expression match

array **mb_ereg_search_init** (string string [, string pattern [, string option]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search_init() sets `string` and `pattern` for multibyte regular expression. These values are used for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). It returns `TRUE` for success, `FALSE` for error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_regs().

mb_ereg_search_getregs (PHP 4 CVS only)

Retrive the result from the last multibyte regular expression match

array **mb_ereg_search_getregs** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search_getregs() returns an array including the sub-string of matched part by last mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). If there are some maches, the

first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns FALSE on error;

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_getpos (PHP 4 CVS only)

Returns start point for next regular expression match

array **mb_ereg_search_getpos** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search_getpos() returns the point to start regular expression match for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). The position is represented by bytes from the head of string.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Anmerkung: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_setpos().

mb_ereg_search_setpos (PHP 4 CVS only)

Set start point of next regular expression match

array **mb_ereg_search_setpos** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

mb_ereg_search_setpos() sets the starting point of match for mb_ereg_search().

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Anmerkung:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

LIII. MCAL Funktionen

MCAL bedeutet Modular Calendar Access Library.

Libmcal ist eine C Bibliothek für den Zugriff auf Kalender. Sie ist sehr modular programmiert, mit austauschbaren Treibern. MCAL ist das Kalendergegenstück zum IMAP Modul für Mailboxen.

Mit Hilfe der mcal-Unterstützung kann ein Kalender-Stream, ähnlich wie ein Mailbox-Stream mit der IMAP-Unterstützung, geöffnet werden. Kalender können lokale Dateispeicher, entfernte ICAP-Server oder andere Formate,von der mcal-Bibliothek unterstützt werden, sein.

Kalendereinträge können aufgerufen, abgefragt und gespeichert werden. Desweiteren werden Kalendertermine (Alarm) und wiederkehrende Termine unterstützt.

Mit libmcal kann auf zentrale Kalenderserver zugegriffen und diese benutzt werden, was die Programmierung einer bestimmten Datenbank oder einer Datei unnötig macht.

Um diese Funktionen benutzen zu können, muß PHP mit der Option `--with-mcal` kompiliert werden. Dies setzt eine installierte mcal-Bibliothek vorraus. Holen Sie sich die aktuelle Version unter http://mcal.chek.com/ und kompilieren und installieren Sie sie.

Die folgenden Konstanten werden definiert, wenn das mcal-Modul verwendet wird: MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRIL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUGT, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, and MCAL_DECEMBER. Die meisten dieser Funktionen benutzen eine interne Ereignisstruktur, die einzigartig für jeden Stream ist. Dies vermindert die Notwendigkeit, große Objekte zwischen Funktionen auszutauschen. Es gibt Hilfsfunktionen für Einstellung, Initialisierung und Abruf der Ereignisstruktur.

mcal_open (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Stellt eine MCAL-Verbindung her

int **mcal_open** (string calendar, string username, string password, string options)

Bringt bei Erfolg einen MCAL-Stream und `FALSE` bei einem Fehler.

mcal_open() öffnet eine MCAL-Verbidnung zu dem angegebenen *`calendar`* Speicher. Wenn der optionale Parameter *`options`* benutzt wird, wird *`options`* auch an die Mailbox weitergegeben. Die interne Ereignisstruktur der Streams wird auch bei der Verbindung initialisiert.

mcal_close (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Schließen eines MCAL-Streams

int **mcal_close** (int mcal_stream, int flags)

Schließt einen offenen mcal-Stream.

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Holt ein Ereignis aus einem Kalender-Stream.

object **mcal_fetch_event** (int mcal_stream, int event_id [, int options])

mcal_fetch_event() holt ein Ereignis aus dem Kalender-Stream, der in *`id`* angegeben wurde.

Zeigt ein Ereignis-Objekt an,welches aus folgendem besteht:

- int id - ID des Ereignisses.
- int public - `TRUE` wenn das Ereignis öffentlich ist, `FALSE` wenn es privat ist.
- string category - Kategorie des Ereignisses.
- string title - Titel des Ereignisses.
- string description - Beschreibung des Ereignisses.
- int alarm - Anzahl der Minuten, bevor das Ereignis einen Alarm bzw. eine Erinnerung sendet.
- object start - Objekt, welches einen Eintrag zu Datum und Uhrzeit enthält.
- object end - Objekt, welches einen Eintrag zu Datum und Uhrzeit enthält.
- int recur_type - Typ des erneuten Auftretens
- int recur_interval - Zeitabstand bis zum erneuten Auftretens
- datetime recur_enddate - Enddatum des erneuten Auftretens
- int recur_data - Daten des erneuten Auftretens

Alle Zeitpunkteinträge bestehen aus einem Objekt, welches folgendes enthält:

- int year - Jahr

- int month - Monat
- int mday - Tag des Monats
- int hour - Stunde
- int min - Minuten
- int sec - Sekunden
- int alarm - Minuten bis zum Zeitpunkt, wenn ein Eintrag einen Alarm sendet

mcal_list_events (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Zeigt eine Liste der Einträge zwischen zwei Zeitpunkten an

array **mcal_list_events** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])

Returns an array of event ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_append_event (PHP 4 >= 4.0.0)

Store a new event into an MCAL calendar

int **mcal_append_event** (int mcal_stream)

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the uid of the newly inserted event.

mcal_store_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Modify an existing event in an MCAL calendar

int **mcal_store_event** (int mcal_stream)

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns TRUE on success and FALSE on error.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Delete an event from an MCAL calendar

int **mcal_delete_event** (int uid)

mcal_delete_event() deletes the calendar event specified by the uid.

Returns TRUE.

mcal_snooze (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Turn off an alarm for an event

int **mcal_snooze** (int uid)

mcal_snooze() turns off an alarm for a calendar event specified by the uid.

Returns TRUE.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return a list of events that has an alarm triggered at the given datetime

array **mcal_list_events** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_event_init (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Initializes a streams global event structure

int **mcal_event_init** (int stream)

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns TRUE.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the category of the streams global event structure

int **mcal_event_set_category** (int stream, string category)

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns TRUE.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the title of the streams global event structure

int **mcal_event_set_title** (int stream, string title)

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns TRUE.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the description of the streams global event structure

int **mcal_event_set_description** (int stream, string description)

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns TRUE.

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the start date and time of the streams global event structure

int **mcal_event_set_start** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns TRUE.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the end date and time of the streams global event structure

int **mcal_event_set_end** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns TRUE.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the alarm of the streams global event structure

int **mcal_event_set_alarm** (int stream, int alarm)

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns TRUE.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the class of the streams global event structure

int **mcal_event_set_class** (int stream, int class)

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns TRUE.

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns if the given year is a leap year or not

int **mcal_is_leap_year** (int year)

mcal_is_leap_year() returns 1 if the given year is a leap year, 1 if not.

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the number of days in the given month

int **mcal_days_in_month** (int month, int leap year)

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_date_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns TRUE if the given year, month, day is a valid date

int **mcal_date_valid** (int year, int month, int day)

mcal_date_valid() Returns TRUE if the given year, month and day is a valid date, FALSE if not.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns TRUE if the given year, month, day is a valid time

int **mcal_time_valid** (int hour, int minutes, int seconds)

mcal_time_valid() Returns TRUE if the given hour, minutes and seconds is a valid time, FALSE if not.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the day of the week of the given date

int **mcal_** (int year, int month, int day)

mcal_day_of_week() returns the day of the week of the given date

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the day of the year of the given date

int **mcal_** (int year, int month, int day)

mcal_day_of_year() returns the day of the year of the given date

mcal_date_compare (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Compares two dates

int **mcal_date_compare** (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day)

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the next recurrence of the event

int **mcal_next_recurrence** (int stream, int weekstart, array next)

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_none** (int stream)

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to MCAL_RECUR_NONE).

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_daily** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_weekly** (int stream, int year, int month, int day, int interval, int weekdays)

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_mday** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_wday** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_yearly** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date .

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns an object containing the current streams event structure

int **mcal_fetch_current_stream_event** (int stream)

mcal_event_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

LIV. Mcrypt Encryption Functions

These functions work using mcrypt (ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt).

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free".

If you linked against libmcrypt 2.4.x, the following additional block algorithms are supported: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT and the following stream ciphers: ENIGMA (crypt), PANAMA, RC4 and WAKE. With libmcrypt 2.4.x another cipher mode is also available; nOFB.

To use it, download libmcrypt-x.x.tar.gz from here (ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt) and follow the included installation instructions. You need to compile PHP with the `--with-mcrypt` parameter to enable this extension. Make sure you compile libmcrypt with the option `--disable-posix-threads`.

Mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. If you linked against libmcrypt-2.2.x, the four important mcrypt commands (mcrypt_cfb(), mcrypt_cbc(), mcrypt_ecb(), and mcrypt_ofb()) can operate in both modes which are named MCRYPT_ENCRYPT and MCRYPT_DECRYPT, respectively.

Beispiel 1. Encrypt an input value with TripleDES under 2.2.x in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o’clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

If you linked against libmcrypt 2.4.x, these functions are still available, but it is recommended that you use the advanced functions.

Beispiel 2. Encrypt an input value with TripleDES under 2.4.x in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o’clock at the secret place.";

$td = mcrypt_module_open (MCRYPT_TripleDES, "", MCRYPT_MODE_ECB, "");
$iv = mcrypt_create_iv (mcrypt_enc_get_iv_size ($td), MCRYPT_RAND);
mcrypt_generic_init ($td, $key, $iv);
$encrypted_data = mcrypt_generic ($td, $input);
mcrypt_generic_end ($td);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x mcrypt can also operate in the block cipher mode nOFB and in STREAM mode. Then there are also constants in the form MCRYPT_MODE_mode for use with several functions. We will outline the normal use for each of these modes. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It's insecure (because it operates in 8bit mode) so it is not recommended to use it.
- nOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the algorithm.
- STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

PHP does not support encrypting/decrypting bit streams currently. As of now, PHP only supports handling of strings.

For a complete list of supported ciphers, see the defines at the end of `mcrypt.h`. The general rule with the mcrypt-2.2.x API is that you can access the cipher from PHP with MCRYPT_ciphername. With the mcrypt-2.4.x API these constants also work, but it is possible to specify the name of the cipher as a string with a call to mcrypt_module_open().

Here is a short list of ciphers which are currently supported by the mcrypt extension. If a cipher is not listed here, but is listed by mcrypt as supported, you can safely assume that this documentation is

outdated.

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt 2.4.x only)
- MCRYPT_ARCFOUR (libmcrypt 2.4.x only)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt 2.4.x only, alias for MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt 2.4.x only)
- MCRYPT_MARS (libmcrypt 2.4.x only, non-free)
- MCRYPT_PANAMA (libmcrypt 2.4.x only)
- MCRYPT_RIJNDAEL_128 (libmcrypt 2.4.x only)
- MCRYPT_RIJNDAEL_192 (libmcrypt 2.4.x only)
- MCRYPT_RIJNDAEL_256 (libmcrypt 2.4.x only)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x only)
- MCRYPT_RC6 (libmcrypt 2.4.x only)
- MCRYPT_RC6_128 (libmcrypt 2.2.x only)
- MCRYPT_RC6_192 (libmcrypt 2.2.x only)
- MCRYPT_RC6_256 (libmcrypt 2.2.x only)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt 2.4.x only)
- MCRYPT_SERPENT (libmcrypt 2.4.x only)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x only)
- MCRYPT_SKIPJACK (libmcrypt 2.4.x only)
- MCRYPT_TEAN (libmcrypt 2.2.x only)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt 2.4.x only)
- MCRYPT_TWOFISH (for older mcrypt 2.x versions, or mcrypt 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions, but not in the 2.4.x

versions)

- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt 2.4.x only)
- MCRYPT_XTEA (libmcrypt 2.4.x only)

You must (in CFB and OFB mode) or can (in CBC mode) supply an initialization vector (IV) to the respective cipher function. The IV must be unique and must be the same when decrypting/encrypting. With data which is stored encrypted, you can take the output of a function of the index under which the data is stored (e.g. the MD5 key of the filename). Alternatively, you can transmit the IV together with the encrypted data (see chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic).

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the name of the specified cipher

string **mcrypt_get_cipher_name** (int cipher)string **mcrypt_get_cipher_name** (string cipher)

mcrypt_get_cipher_name() is used to get the name of the specified cipher.

mcrypt_get_cipher_name() takes the cipher number as an argument (libmcrypt 2.2.x) or takes the cipher name as an argument (libmcrypt 2.4.x) and returns the name of the cipher or `FALSE`, if the cipher does not exist.

Beispiel 1. mcrypt_get_cipher_name() Example

```
<?php
$cipher = MCRYPT_TripleDES;

print mcrypt_get_cipher_name ($cipher);
?>
```

The above example will produce:

```
3DES
```

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the block size of the specified cipher

int **mcrypt_get_block_size** (int cipher)int **mcrypt_get_block_size** (string cipher, string module)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_get_block_size() is used to get the size of a block of the specified *`cipher`*.

mcrypt_get_block_size() takes one or two arguments, the *`cipher`* and *`module`*, and returns the size in bytes.

See also: mcrypt_get_key_size().

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the key size of the specified cipher

int **mcrypt_get_key_size** (int cipher)int **mcrypt_get_key_size** (string cipher, string module)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_get_key_size() is used to get the size of a key of the specified *cipher*.

mcrypt_get_key_size() takes one or two arguments, the *cipher* and *module*, and returns the size in bytes.

See also: mcrypt_get_block_size().

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Create an initialization vector (IV) from a random source

string **mcrypt_create_iv** (int size, int source)

mcrypt_create_iv() is used to create an IV.

mcrypt_create_iv() takes two arguments, *size* determines the size of the IV, *source* specifies the source of the IV.

The source can be MCRYPT_RAND (system random number generator), MCRYPT_DEV_RANDOM (read data from /dev/random) and MCRYPT_DEV_URANDOM (read data from /dev/urandom). If you use MCRYPT_RAND, make sure to call srand() before to initialize the random number generator.

Beispiel 1. mcrypt_create_iv() example

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size ($cipher);
$iv = mcrypt_create_iv ($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in CBC mode

string **mcrypt_cbc** (int cipher, string key, string data, int mode [, string iv])string **mcrypt_cbc** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_cbc() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CBC cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the optional initialization vector.

See also: mcrypt_cfb(), mcrypt_ecb(), and mcrypt_ofb().

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in CFB mode

string **mcrypt_cfb** (int cipher, string key, string data, int mode, string iv)string **mcrypt_cfb** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_cfb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CFB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the initialization vector.

See also: mcrypt_cbc(), mcrypt_ecb(), and mcrypt_ofb().

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in ECB mode

string **mcrypt_ecb** (int cipher, string key, string data, int mode)string **mcrypt_ecb** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_ecb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in ECB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

See also: mcrypt_cbc(), mcrypt_cfb(), and mcrypt_ofb().

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in OFB mode

string **mcrypt_ofb** (int cipher, string key, string data, int mode, string iv)string **mcrypt_ofb** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_ofb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in OFB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the initialization vector.

See also: mcrypt_cbc(), mcrypt_cfb(), and mcrypt_ecb().

mcrypt_list_algorithms (PHP 4 >= 4.0.2)

Get an array of all supported ciphers

array **mcrypt_list_algorithms** ([string lib_dir])

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the

lib_dir. **mcrypt_list_algorithms()** takes as optional parameter a directory which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir php.ini directive is used.

Beispiel 1. mcrypt_list_algorithms() Example

```
<?php
$algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");

foreach ($algorithms as $cipher) {
  echo $cipher."/n";
}
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes (PHP 4 >= 4.0.2)

Get an array of all supported modes

array **mcrypt_list_modes** ([string lib_dir])

mcrypt_list_modes() is used to get an array of all supported modes in the *lib_dir*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir php.ini directive is used.

Beispiel 1. mcrypt_list_modes() Example

```
<?php
$modes = mcrypt_list_modes ();

foreach ($modes as $mode) {
  echo "$mode <br/>";
}
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV belonging to a specific cipher/mode combination

int **mcrypt_get_iv_size** (string cipher, string mode)int **mcrypt_get_iv_size** (resource td)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x.

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns `FALSE`. If the IV is ignored in the specified cipher/mode combination zero is returned.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

Td is the algorithm specified.

mcrypt_encrypt (PHP 4 >= 4.0.2)

Encrypts plaintext with given parameters

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv])

mcrypt_encrypt() encrypts the data and returns the encrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '\0'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

Data is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'. The returned crypttext can be larger that the size of the data that is given by *data*.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

Beispiel 1. mcrypt_encrypt() Example

```
<?php
$iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MCRYPT
$key = "This is a very secret key";
$text = "Meet me at 11 o'clock behind the monument.";
echo strlen ($text)."\n";

$crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
echo strlen ($crypttext)."\n";
?>
```

The above example will print out:

```
42
64
```

mcrypt_decrypt (PHP 4 >= 4.0.2)

Decrypts crypttext with given parameters

string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv])

mcrypt_decrypt() decrypts the data and returns the unencrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '\0'.

Data is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

mcrypt_module_open (PHP 4 >= 4.0.2)

This function opens the module of the algorithm and the mode to be used

resource **mcrypt_module_open** (string algorithm, string algorithm_directory, string mode, string mode_directory)

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, eg "twofish" or is one of the MCRYPT_ciphername constants. The library is closed by calling mcrypt_module_close(), but there is no need to call that function if mcrypt_generic_end() is called. Normally it returns an encryption descriptor, or FALSE on error.

The *algorithm_directory* and *mode_directory* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string ("""), the value set by the *mcrypt.algorithms_dir* or *mcrypt.modes_dir* ini-directive is used. When these are not set, the default directory are used that are compiled in into libmcrypt (usally /usr/local/lib/libmcrypt).

Beispiel 1. mcrypt_module_open() Example

```
<?php
$td = mcrypt_module_open (MCRYPT_DES, "", MCRYPT_MODE_ECB, "/usr/lib/mcrypt-
modes");
?>
```

The above example will try to open the DES cipher from the default directory and the EBC mode from the directory /usr/lib/mcrypt-modes.

mcrypt_module_close (PHP 4 >= 4.0.2)

Free the descriptor td

bool **mcrypt_module_close** (resource td)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mcrypt_generic_deinit (PHP 4 >= 4.1.1)

This function terminates encrypt specified by the descriptor td

bool **mcrypt_generic_deinit** (resource td)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mcrypt_generic_init (PHP 4 >= 4.0.2)

This function initializes all buffers needed for encryption

int **mcrypt_generic_init** (resource td, string key, string iv)

The maximum length of the key should be the one obtained by calling mcrypt_enc_get_key_size() and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling mcrypt_enc_get_iv_size(). IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended. The function returns a negative value on error.

You need to call this function before every mcrypt_generic() or mdecrypt_generic().

mcrypt_generic (PHP 4 >= 4.0.2)

This function encrypts data

string **mcrypt_generic** (resource td, string data)

This function encrypts data. The data is padded with "\0" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

mdecrypt_generic (PHP 4 >= 4.0.2)

This function decrypts data

string **mdecrypt_generic** (resource td, string data)

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Beispiel 1. mdecrypt_generic() Example

```
<?php
$iv_size = mcrypt_enc_get_iv_size ($td));
$iv = @mcrypt_create_iv ($iv_size, MCRYPT_RAND);

if (@mcrypt_generic_init ($td, $key, $iv) != -1)
{
    $c_t = mcrypt_generic ($td, $plain_text);
    @mcrypt_generic_init ($td, $key, $iv);
    $p_t = mdecrypt_generic ($td, $c_t);
}
if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0)
    echo "ok";
else
    echo "error";
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption.

mcrypt_generic_end (PHP 4 >= 4.0.2)

This function terminates encryption

bool **mcrypt_generic_end** (resource td)

This function terminates encryption specified by the encryption descriptor (td). Actually it clears all buffers, and closes all the modules used. Returns `FALSE` on error, or `TRUE` on succes.

mcrypt_enc_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the opened module

int **mcrypt_enc_self_test** (resource td)

This function runs the self test on the algorithm specified by the descriptor td. If the self test succeeds it returns zero. In case of an error, it returns 1.

mcrypt_enc_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Checks whether the encryption of the opened mode works on blocks

int **mcrypt_enc_is_block_algorithm_mode** (resource td)

This function returns 1 if the mode is for use with block algorithms, otherwise it returns 0. (eg. 0 for stream, and 1 for cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm (PHP 4 >= 4.0.2)

Checks whether the algorithm of the opened mode is a block algorithm

int **mcrypt_enc_is_block_algorithm** (resource td)

This function returns 1 if the algorithm is a block algorithm, or 0 if it is a stream algorithm.

mcrypt_enc_is_block_mode (PHP 4 >= 4.0.2)

Checks whether the opened mode outputs blocks

int **mcrypt_enc_is_block_mode** (resource td)

This function returns 1 if the mode outputs blocks of bytes or 0 if it outputs bytes. (eg. 1 for cbc and ecb, and 0 for cfb and stream).

mcrypt_enc_get_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the opened algorithm

int **mcrypt_enc_get_block_size** (resource td)

This function returns the block size of the algorithm specified by the encryption descriptor td in bytes.

mcrypt_enc_get_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_enc_get_key_size** (resource td)

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor td in bytes.

mcrypt_enc_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_enc_get_supported_key_sizes** (resource td)

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and mcrypt_enc_get_key_size() are supported by the algorithm.

mcrypt_enc_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV of the opened algorithm

int **mcrypt_enc_get_iv_size** (resource td)

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns ’0’ then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_algorithms_name (PHP 4 >= 4.0.2)

Returns the name of the opened algorithm

string **mcrypt_enc_get_algorithms_name** (resource td)

This function returns the name of the algorithm.

mcrypt_enc_get_modes_name (PHP 4 >= 4.0.2)

Returns the name of the opened mode

string **mcrypt_enc_get_modes_name** (resource td)

This function returns the name of the mode.

mcrypt_module_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the specified module

bool **mcrypt_module_self_test** (string algorithm [, string lib_dir])

This function runs the self test on the algorithm specified. The optional *lib_dir* parameter can contain the location of where the algorithm module is on the system.

The function returns TRUE if the self test succeeds, or FALSE when if fails.

mcrypt_module_is_block_algorithm_mode (PHP 4 >= 4.0.2)

This function returns if the the specified module is a block algorithm or not

bool **mcrypt_module_is_block_algorithm_mode** (string mode [, string lib_dir])

This function returns TRUE if the mode is for use with block algorithms, otherwise it returns 0. (eg. 0 for stream, and 1 for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm (PHP 4 >= 4.0.2)

This function checks whether the specified algorithm is a block algorithm

bool **mcrypt_module_is_block_algorithm** (string algorithm [, string lib_dir])

This function returns TRUE if the specified algorithm is a block algorithm, or FALSE is it is a stream algorithm. The optional *lib_dir* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode (PHP 4 >= 4.0.2)

This function returns if the the specified mode outputs blocks or not

bool **mcrypt_module_is_block_mode** (string mode [, string lib_dir])

This function returns TRUE if the mode outputs blocks of bytes or FALSE if it outputs just bytes. (eg. 1 for cbc and ecb, and 0 for cfb and stream). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the specified algorithm

int **mcrypt_module_get_algo_block_size** (string algorithm [, string lib_dir])

This function returns the block size of the algorithm specified in bytes. The optional `lib_dir` parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_module_get_algo_key_size** (string algorithm [, string lib_dir])

This function returns the maximum supported key size of the algorithm specified in bytes. The optional `lib_dir` parameter can contain the location where the mode module is on the system.

mcrypt_module_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_module_get_supported_key_sizes** (string algorithm [, string lib_dir])

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and mcrypt_module_get_algo_key_size() are supported by the algorithm. The optional `lib_dir` parameter can contain the location where the mode module is on the system.

LV. Hash Funktionen

Diese Funktionen arbeiten zusammen mit mhash (http://mhash.sourceforge.net/).

Dies ist eine Schnittstelle zu der Mhash Bibliothek. Mhash unterstützt eine breite Auswahl an Hash Algorithmen, wie MD5, SHA1, GOST, und viele andere.

Um sie zu verwenden, muss man die Mhash Distribution von der Website (http://mhash.sourceforge.net/) downloaden und der beigefügten Instalationsanleitung folgen. PHP muss mit der Option `--with-mhash` kompiliert werden um diese Erweiterung zu aktivieren.

Mhash wird verwendet um z.B. Checksummen und Mehr zu erstellen.

Beispiel 1. Berechne den MD5 Extrakt und gib es als Hex aus

```
<?php
$input = "what do ya want for nothing?";
$hash = mhash (MHASH_MD5, $input);
print "Der Hash ist ".bin2hex ($hash)."\n&lt;br>";
$hash = mhash (MHASH_MD5, $input, "Jefe");
print "Der hmac ist ".bin2hex ($hash)."\n&lt;br>";
?>
```

Die Ausgabe ist:

```
Der Hash ist d03cb659cbf9192dcd066272249f8412
Der hmac ist 750c783e6ab0b503eaa86e310a5db738
```

Eine komplette Liste aller unterstützen Hashes ist in der Dokumentation von Mhash enthalten. Die allgemeine Verwendung ist, dass man einen Hash Algorithmus von PHP mit MHASH_HASHNAME erreicht. Z.B., um HAVAL zu erreichen benutzt man die PHP Konstante MHASH_HAVAL.

Hier ist eine Liste von Hashs, die momentan von Mhash unterstützt werden. Wenn ein Hash ist hier nicht aufgeführt ist, aber von Mhash, laut Dokumentation, unterstützt wird, ist diese Dokumentation

als veraltet anzusehen.

- MHASH_MD5
- MHASH_SHA1
- MHASH_HAVAL
- MHASH_RIPEMD160
- MHASH_RIPEMD128
- MHASH_SNEFRU
- MHASH_TIGER
- MHASH_GOST
- MHASH_CRC32
- MHASH_CRC32B

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Gibt den Namen eines Hashs zurück

string **mhash_get_hash_name** (int hash)

mhash_get_hash_name() wird verwendet um den Namen eines Hashes zu erfahren.

mhash_get_hash_name() nimmt die Hash Id als Argument und gibt den Namen des Hashs zurück oder `FALSE`, wenn der Hash nicht existiert.

Beispiel 1. Mhash_get_hash_name Beispiel

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name ($hash);
?>
```

Das Beispiel oben gibt aus:

```
MD5
```

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Gibt die Blockgroesse von dem übergebenem Hash zurück

int **mhash_get_block_size** (int hash)

mhash_get_block_size() gibt die Groesse des übergebenen *`Hashs`*.

mhash_get_block_size() nimmt ein Argument, den *`Hash`* und gibt die Grösse in Byte oder `FALSE`, wenn *`Hash`* nicht existiert.

mhash_count (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Gibt den höchstmöglichen Hash zurück

int **mhash_count** (void)

mhash_count() gibt die höchsten verfügbaren Hash Id zurück. Hashes sind von 0 bis zu der Hash Id nummeriert.

Beispiel 1. Alle Hashes durchlaufen

```
<?php

$nr = mhash_count();

for ($i = 0; $i <= $nr; $i++) {
    echo sprintf ("Die Blockgrösse von %s ist %d\n",
        mhash_get_hash_name ($i),
        mhash_get_block_size ($i));
}
?>
```

mhash (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Hash berechnen

string **mhash** (int hash, string data)

mhash() verwendet eine Hash Funktion abhängig von *hash* auf *data* und gibt den resultierenden Hash zurück (wird auch Digest genannt).

LVI. Microsoft SQL Server Funktionen

Die MSSQL-Erweiterung steht ihnen nur auf Win32-Ssystemen zur Verfügung. Sie können die Sybase-Erweiterung benutzen, um von anderen Plattformen aus eine Verbindung zu MS SQL-Datenbanken aufzubauen.

Diese Funktionen ermöglichen Ihnen den Zugriff auf MSSQL-Datenbanken. Für die Erweiterung müssen die MS SQL Client Tools auf dem System installiert sein auf dem PHP installiert ist. Die Client Tools können entweder von der MS SQL Server-CD oder durch Kopieren von ntwdblib.dll von \winnt\system32 auf dem Server nach \winnt\system32 auf dem PHP-Rechner installiert werden. Das Kopieren von ntwdblib.dll unterstützt nur den Zugriff. Für die Konfiguration des Clients ist eine Installation aller Tools nötig.

Die MSSQL-Erweiterung wird durch den Eintrag extension=php_mssql.dll in die php.ini aktiviert.

mssql_close (PHP 3, PHP 4 >= 4.0.0)

Schließt die Verbindung zum MS SQL Server

int **mssql_close** ([int Verbindungs-Kennung])

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück.

mssql_close() schließt die Verbindung zu der MS SQL Server Datenbank, die zur angegebenen *`Verbindungs-Kennung`* gehört. Wenn keine Verbindungs-Kennung angegeben wird, wird von der zuletzt geöffneten Verbindung ausgegangen.

Beachten Sie, dass dies normalerweise nicht nötig ist, weil nichtpersistente offene Verbindungen automatisch nach Ausführung des PHP-Skripts geschlossen werden.

mssql_close() schließt keine mittels mssql_pconnect() aufgebauten persistenten Verbindungen.

Siehe auch: mssql_connect() und mssql_pconnect().

mssql_connect (PHP 3, PHP 4 >= 4.0.0)

Baut eine Verbindung zum MS SQL Server auf

int **mssql_connect** ([string Servername [, string Benutzername [, string Passwort]]])

Gibt bei Erfolg eine positive MS SQL Verbindungs-Kennung oder bei Auftreten eines Fehlers `FALSE` zurück.

mssql_connect() baut eine Verbindung zu einem MS SQL Server auf. Das Argument *`Servername`* muss ein gültiger Servername sein, der in der 'interfaces'-Datei definiert ist.

Falls **mssql_connect()** zum zweiten Mal mit den selben Argumenten aufgerufen wird, wird keine neue Verbindung aufgebaut, sondern stattdessen die Verbindungs-Kennung der bereits bestehenden Verbindung zurückgegeben.

Die Verbindung zum Server wird geschlossen, sobald die Ausführung des Skripts abgeschlossen ist, wenn sie nicht schon vorher durch einen expliziten Aufruf von mssql_close() geschlossen wurde.

Siehe auch: mssql_pconnect() und mssql_close().

mssql_data_seek (PHP 3, PHP 4 >= 4.0.0)

Bewegt den internen Datensatz-Zeiger

int **mssql_data_seek** (int Ergebnis-Kennung, int Zeilennummer)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück.

mssql_data_seek() bewegt den internen Datensatz-Zeiger des mit der angegebenen *`Ergebnis-Kennung`* verbundenen Ergebnisses zur angegebenen *`Zeilennummer`*. Beim nächsten Aufruf von mssql_fetch_row() wird nun dieser Datensatz zurückgeben.

Siehe auch: **mssql_data_seek()**.

mssql_fetch_array (PHP 3, PHP 4 >= 4.0.0)

Liefert einen Datensatz als Array

int **mssql_fetch_array** (int Ergebnis)

Gibt ein Array mit dem gelieferten Datensatz zurück oder FALSE, wenn es keine weiteren Datensätze gibt.

mssql_fetch_array() ist die erweiterte Version von mssql_fetch_row(). Zusätzlich zum Speichern der Daten in den numerischen Indizes des Ergebnisarrays, speichert sie die Daten unter Verwendung der Feldnamen als Schlüssel auch in assoziativen Indizes.

Wichtige Anmerkung: die Verwendung von **mssql_fetch_array()** ist nur unwesentlich langsamer als die Verwendung von mssql_fetch_row(), bietet aber wesentlich mehr.

Weitere Einzelheiten, siehe auch: mssql_fetch_row().

mssql_fetch_field (PHP 3, PHP 4 >= 4.0.0)

Liefert Informationen über ein Feld

object **mssql_fetch_field** (int Ergebnis [, int Feld-Offset])

Gibt ein Objekt mit Informationen über ein Feld zurück.

mssql_fetch_field() wird verwendet, um im Ergebnis einer Anfrage Informationen über Felder zu erhalten. Wurde kein *Feld-Offset* angegeben, wird das nächste Feld, das noch nicht abgefragt wurde, geliefert.

Die Eigenschaften dieses Objekts sind:

- name - Feldname. Wenn das Feld das Ergebnis einer Funktion ist, wird diese Eigenschaft auf computed#N gesetzt, wobei #N eine fortlaufende Nummer ist.
- column_source - die Tabelle, zu der das Feld gehört
- max_length - maximale Feldbreite
- numeric - 1, wenn es sich um ein numerisches Feld handelt

Siehe auch: mssql_field_seek().

mssql_fetch_object (PHP 3, PHP 4 >= 4.0.0)

Liefert einen Datensatz als Objekt

int **mssql_fetch_object** (int Ergebnis)

Gibt ein Objekt mit den Eigenschaften des gelieferten Datensatzes zurück oder FALSE, wenn es keine weiteren Datensätze gibt.

mssql_fetch_object() ist mssql_fetch_array() ähnlich, aber es besteht ein Unterschied: statt eines Arrays wird ein Objekt zurückgegeben. Indirekt bedeutet das, dass nur über die Feldnamen auf die

Daten zugegriffen werden kann, nicht aber über ihre Offsets (Zahlen sind als Namen von Eigenschaften nicht erlaubt).

Was die Geschwindigkeit betrifft, ist diese Funktion mit mssql_fetch_array() identisch und fast genauso schnell wie mssql_fetch_row() (der Unterschied ist unbedeutend).

Siehe auch: mssql_fetch-array() und mssql_fetch-row().

mssql_fetch_row (PHP 3, PHP 4 >= 4.0.0)

Liefert einen Datensatz als indiziertes Array

array **mssql_fetch_row** (int Ergebnis)

Gibt ein Array mit dem gelieferten Datensatz zurück oder `FALSE`, wenn es keinen weiteren Datensatz gibt.

mssql_fetch_row() liefert einen Datensatz von dem Ergebnis, das zur angegebenen *`Ergebnis`*-Kennung gehört. Der Datensatz wird als Array zurückgegeben. Jedes Feld des Ergebnisses wird in einem Array-Offset gespeichert, der bei Offset 0 beginnt.

Ein nachfolgender Aufruf von **mssql_fetch_rows()** gibt den nächsten Datensatz aus dem Ergebnis zurück oder `FALSE`, wenn es keine weiteren Datensätze gibt.

Siehe auch: mssql_fetch_array(), mssql_fetch_object(), mssql_data_seek(), **mssql_fetch_lengths()** und mssql_result().

mssql_field_length (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Liefert die Länge eines Feldes

int **mssql_field_length** (int Ergebnis [, int Offset])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_field_name (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Liefert den Namen eines Feldes

int **mssql_field_name** (int Ergebnis [, int Offset])

mssql_field_seek (PHP 3, PHP 4 >= 4.0.0)

Setzt einen Feld-Offset

int **mssql_field_seek** (int Ergebnis, int Feld-Offset)

Sucht bis zum angegebenen `Feld-Offset`. Wenn der nächste Aufruf von mssql_fetch_field() keinen Feld-Offset enthält, wird dieses Feld zurückgegeben.

Siehe auch: mssql_fetch_field().

mssql_field_type (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Liefert den Typ eines Feldes

string **mssql_field_type** (int Ergebnis [, int Offset])

mssql_free_result (PHP 3, PHP 4 >= 4.0.0)

Gibt den Ergebnisspeicher frei

int **mssql_free_result** (int Ergebnis)

mssql_free_result() brauchen Sie nur aufrufen, wenn Sie sich über zu großen Speicherbedarf zur Laufzeit ihres Skripts Sorgen machen. Der Ergebnisspeicher wird automatisch bei Beendigung des Skripts freigegeben. Sie können **mssql_free_result()** mit der `Ergebnis`-Kennung als Argument aufrufen und damit den zugehörigen Ergebnisspeicher freigeben.

mssql_get_last_message (PHP 3, PHP 4 >= 4.0.0)

Gibt die letzte Meldung des Servers zurück (über min_message_severity?)

string **mssql_get_last_message** (void)

mssql_min_error_severity (PHP 3, PHP 4 >= 4.0.0)

Setzt die untere Fehlerschwelle

void **mssql_min_error_severity** (int Schwellenwert)

mssql_min_message_severity (PHP 3, PHP 4 >= 4.0.0)

Setzt die untere Schwelle für Meldungen

void **mssql_min_message_severity** (int Schwellenwert)

mssql_next_result (PHP 4 >= 4.0.5)

Bewegt den internen Ergebnis-Zeiger zum nächsten Ergebnis

bool **mssql_next_result** (int Ergebnis-ID)

Wird mehr als eine SQL-Anweisung an den Server geschickt oder eine Stored Procedure mit mehreren Ergebnissen ausgeführt, dann liefert der Server mehrere Ergebnisse. Diese Funktion prüft, ob vom Server noch weitere Ergebnisse angeboten werden. Wenn es ein weiteres Ergebnis gibt, gibt sie das existierende Ergebnis frei und bereitet das nächste Ergebnis vor. Liegt ein weiteres Ergebnis vor, gibt die Funktion `TRUE`, andernfalls `FALSE` zurück.

Beispiel 1. mssql_next_result() Beispiel

```
<?php
    $link = mssql_connect ("localhost", "benutzer", "geheim");
    mssql_select_db("MyDB", $link);
    $SQL = "Select * from table1 select * from table2";
    $rs = mssql_query($SQL, $link);
    do {
        while ($row = mssql_fetch_row($rs)) {
        }
    } while (mssql_next_result($rs));
    mssql_free_result($rs);
    mssql_close ($link);
?>
```

mssql_num_fields (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Felder eines Ergebnisses

int **mssql_num_fields** (int Ergebnis)

mssql_num_fields() Gibt die Anzahl der Felder eines Ergebnisses zurück.

Siehe auch: **mssql_db_query()**, mssql_query(), mssql_fetch_field() und mssql_num_rows().

mssql_num_rows (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Datensätze eines Ergebnisses

int **mssql_num_rows** (string Ergebnis)

mssql_num_rows() Gibt die Anzahl der Datensätze eines Ergebnisses zurück.

Siehe auch: **mssql_db_query()**, mssql_query() und mssql_fetch_row().

mssql_pconnect (PHP 3, PHP 4 >= 4.0.0)

Baut eine persistente MS SQL Verbindung auf

int **mssql_pconnect** ([string Servername [, string Benutzername [, string Passwort]]])

Gibt bei Erfolg eine positive persistente MS SQL Verbindungs-Kennung oder, bei Auftreten eines Fehlers, `FALSE` zurück.

mssql_pconnect() verhält sich, von zwei wesentlichen Unterschieden abgesehen, fast genauso wie mssql_connect().

Erstens: Die Funktion wird beim Verbindungsaufbau zunächst versuchen, eine (persistente) Verbindung zu finden, die unter dem selben Host, Benutzername und Passwort geöffnet wurde. Wenn eine solche gefunden wird, gibt sie deren Kennung zurück, statt eine neue Verbindung aufzubauen.

Zweitens: die Verbindung zum SQL Server wird bei Beendigung des Skripts nicht geschlossen. Stattdessen bleibt die Verbindung für eine spätere Verwendung bestehen (mssql_close() schließt keine Verbindungen, die mittels **mssql_pconnect()** aufgebaut wurden).

Eine solche Verbindung wird deshalb als 'persistent' bezeichnet.

mssql_query (PHP 3, PHP 4 >= 4.0.0)

Sendet eine MS SQL Anfrage

int **mssql_query** (string Anfrage [, int Verbindungs-Kennung])

Gibt bei Erfolg eine positive Ergebnis-Kennung oder, bei Auftreten eines Fehlers, `FALSE` zurück.

mssql_query() sendet eine Anfrage an die gerade aktive Datenbank auf dem Server, die zur angegebenen *`Verbindungs-Kennung`* gehört. Wenn keine Verbindungs-Kennung angegeben wird, wird von der zuletzt geöffneten Verbindung ausgegangen. Wenn es keine offene Verbindung gibt, versucht die Funktion eine Verbindung aufzubauen, als wäre mssql_connect() aufgerufen worden, und diese zu benutzen.

Siehe auch: **mssql_db_query()**, mssql_select_db() und mssql_connect().

mssql_result (PHP 3, PHP 4 >= 4.0.0)

Liefert die bei einer Abfrage gefundenen Daten

int **mssql_result** (int Ergebnis, int i, mixed Feld)

mssql_result() gibt den Inhalt einer Zelle des angegebenen MS SQL Ergebnisdatensatzes zurück. Das Argument *`Feld`* kann der Feld-Offset, der Feldname oder vom Format tabellenname.feldname sein. Wurde ein Alias angegeben ('select foo as bar from...'), wird statt des Feldnamens dieser Alias verwendet.

Wenn Sie mit einer großen Anzahl von Ergebnisdatensätzen arbeiten, sollten Sie in Erwägung ziehen, eine der Funktionen zu verwenden, die komplette Datensätze liefern (siehe unten). Diese Funktionen sind VIEL schneller als **mssql_result()**, weil sie bei einem Funktionsaufruf den Inhalt mehrerer Datenzellen liefern. Beachten Sie auch, dass die Angabe eines numerischen Offsets als Feldargument viel schneller ist, als die Angabe eines Feldnamens oder des Arguments tabellenname.feldname.

Empfohlene leistungsfähige Alternativen: mssql_fetch_row(), mssql_fetch_array() und mssql_fetch_object().

mssql_select_db (PHP 3, PHP 4 >= 4.0.0)

Wählt eine MS SQL Datenbank aus

int **mssql_select_db** (string Datenbankname [, int Verbindungs-Kennung])

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück.

mssql_select_db() setzt auf dem Server die aktuelle Datenbank, auf die sich die angegebene *`Verbindungs-Kennung`* bezieht. Wurde keine Verbindungs-Kennung angegeben, wird die zuletzt geöffnete Verbindung angenommen. Wenn es keine offene Verbindung gibt, versucht die Funktion eine Verbindung aufzubauen, als wäre mssql_connect() aufgerufen worden, und diese zu benutzen.

Jeder folgende Aufruf von mssql_query() bezieht sich auf die geöffnete Datenbank.

Siehe auch: mssql_connect(), mssql_pconnect() und mssql_query()

mssql_bind (PHP 4 >= 4.1.0)

Fügt einer Stored Procedure oder einer Remote Stored Procedure einen Parameter hinzu

int **mssql_bind** (int stmt, string param_name, mixed var, int type [, int is_output [, int is_null [, int maxlen]]])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_execute (PHP 4 >= 4.1.0)

Führt eine Stored Procedure in einer MS SQL-Datenbank aus

int **mssql_execute** (int stmt)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_fetch_assoc (PHP 4 CVS only)

Liefert ein assoziatives Array des aktuellen Datensatzes aus dem Ergebnis, das durch die result_id bestimmt ist

array **mssql_fetch_assoc** (int result_id [, int result_type])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_fetch_batch (PHP 4 >= 4.0.4)

Liefert den nächsten Stapel von Datensätzen

int **mssql_fetch_batch** (string result_index)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_guid_string (PHP 4 >= 4.1.0)

Wandelt eine 16 byte binary GUID in eine Zeichenkette um

string **mssql_guid_string** (string binary [, int short_format])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_init (PHP 4 >= 4.1.0)

Initialisiert eine Stored Procedure oder eine Remote Stored Procedure

int **mssql_init** (string sp_name [, int conn_id])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

mssql_rows_affected (PHP 4 >= 4.0.4)

Liefert die Anzahl der von einer Anfrage betroffenen Datensätze

int **mssql_rows_affected** (int conn_id)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LVII. Ming functions for Flash

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

Introduction

Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms--the only thing that's missing is sound events.

Ming is not an acronym.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know ming@opaque.net (mailto:ming@opaque.net).

Ming was added in PHP 4.0.5.

Installation

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page : http://www.opaque.net/ming/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build `libming.so` and install it into `/usr/lib/`, and copy `ming.h` into `/usr/include/`. Edit the `PREFIX=` line in the `Makefile` to change the installation directory.

built into php (unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
```

Build and install php as usual, Restart web server if necessary

built into php (unix)

download `php_ming.so.gz`. uncompress it and copy it to your php modules directory. (you can find your php module directory by running **php-config --extension-dir**). Now either just add `extension=php_ming.so` to your `php.ini` file, or put `dl('php_ming.so');` at the head of all of your Ming scripts.

How to use Ming

Ming introduces 13 new objects in PHP, all with matching methods and attributes. To use them, you need to know about objects.

- swfmovie().
- swfshape().
- swfdisplayitem().
- swfgradient().
- swfbitmap().
- swffill().
- swfmorph().
- swftext().
- swffont().
- swftextfield().
- swfsprite().
- swfbutton().
- swfaction().

ming_setcubicthreshold (PHP 4 >= 4.0.5)

Set cubic threshold (?)

void **ming_setcubicthreshold** (int threshold)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ming_setscale (PHP 4 >= 4.0.5)

Set scale (?)

void **ming_setscale** (int scale)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

ming_useswfversion (PHP 4 CVS only)

Use SWF version (?)

void **ming_useswfversion** (int version)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

swfbutton_keypress (PHP 4 >= 4.0.5)

Returns the action flag for keyPress(char)

int **swfbutton_keypress** (string str)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

SWFMovie (PHP 4 >= 4.0.5)

Creates a new movie object, representing an SWF version 4 movie.

new **swfmovie** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** and **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFMovie->output (unknown)

Dumps your lovingly prepared movie out.

void **swfmovie->output** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

See also **swfmovie->save()**.

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, swfaction()... Any example will use this method.

SWFMovie->save (unknown)

Saves your movie in a file.

void **swfmovie->save** (string filename)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->save() saves your movie to the file named `filename`.

See also **output()**.

SWFMovie->add (unknown)

Adds any type of data to a movie.

void **swfmovie->add** (ressource instance)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->add() adds `instance` to the current movie. `instance` is any type of data : Shapes, text, fonts, etc. must all be add'ed to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **SWFDisplayItem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->remove (unknown)

Removes the object instance from the display list.

void **swfmovie->remove** (resource instance)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->remove() removes the object instance `instance` from the display list.

See also **swfmovie->add()**.

SWFMovie->setbackground (unknown)

Sets the background color.

void **swfmovie->setbackground** (int red, int green, int blue)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the html background show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setrate (unknown)

Sets the animation's frame rate.

void **swfmovie->setrate** (int rate)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->setrate() sets the frame rate to `rate`, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->setdimension (unknown)

Sets the movie's width and height.

void **swfmovie->setdimension** (int width, int height)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->setdimension() sets the movie's width to `width` and height to `height`.

SWFMovie->setframes (unknown)

Sets the total number of frames in the animation.

void **swfmovie->setframes** (string numberofframes)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->setframes() sets the total number of frames in the animation to `numberofframes`.

SWFMovie->nextframe (unknown)

Moves to the next frame of the animation.

void **swfmovie->nextframe** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->setframes() moves to the next frame of the animation.

SWFMovie->streammp3 (unknown)

Streams a MP3 file.

void **swfmovie->streammp3** (string mp3FileName)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmovie->streammp3() streams the mp3 file *`mp3FileName`*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **SWFShape->addJpegFill()**, this isn't a stable function- we'll probably need to make a separate SWFSound object to contain sound types.

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don't tell the RIAA.

Beispiel 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
  // use your own MP3

  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem (unknown)

Creates a new displayitem object.

new **swfdisplayitem** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto()** **swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth()** **swfdisplayitem->remove()**, **swfdisplayitem->setname()** **swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** and **swfdisplayitem->multcolor()**.

SWFDisplayItem->moveTo (unknown)

Moves object in global coordinates.

void **swfdisplayitem->moveto** (int x, int y)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->moveto() moves the current object to (x,y) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->move (unknown)

Moves object in relative coordinates.

void **swfdisplayitem->move** (int dx, int dy)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->move() moves the current object by (*dx*,*dy*) from its current position.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->scaleTo (unknown)

Scales the object in global coordinates.

void **swfdisplayitem->scaleto** (int x, int y)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->scaleto() scales the current object to (*x*,*y*) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->scale (unknown)

Scales the object in relative coordinates.

void **swfdisplayitem->scale** (int dx, int dy)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->rotateTo (unknown)

Rotates the object in global coordinates.

void **swfdisplayitem->rotateto** (float degrees)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->rotateto() set the current object rotation to `degrees` degrees in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.

Beispiel 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // functions with huge numbers of arbitrary
  // arguments are always a good idea!  Really!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // we can add properties just like a normal php var,
    // as long as the names aren't already used.
    // e.g., we can't set $i->scale, because that's a function

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // but the changes are local to the function, so we have to
```

```
    // return the changed object.  kinda weird..

    return $i;
  }

  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);

    return $i;
  }

  // see?  it sure paid off in legibility:

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->Rotate (unknown)

Rotates in relative coordinates.

void **swfdisplayitem->rotate** (float ddegrees)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->rotate() rotates the current object by `ddegrees` degrees from its current rotation.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->skewXTo (unknown)

Sets the X-skew.

void **swfdisplayitem->skewxto** (float degrees)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->skewxto() sets the x-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewX (unknown)

Sets the X-skew.

void **swfdisplayitem->skewx** (float ddegrees)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->skewx() adds `ddegrees` to current x-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewYTo (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewyto** (float degrees)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->skewyto() sets the y-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewY (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewy** (float ddegrees)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->skewy() adds `ddegrees` to current y-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->setDepth (unknown)

Sets z-order

void **swfdisplayitem->setdepth** (float depth)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->rotate() sets the object's z-order to `depth`. Depth defaults to the order in which instances are created (by add'ing a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->remove (unknown)

Removes the object from the movie

void **swfdisplayitem->remove** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->setName (unknown)

Sets the object's name

void **swfdisplayitem->setname** (string name)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->setname() sets the object's name to *name*, for targetting with action script. Only useful on sprites.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio (unknown)

Sets the object's ratio.

void **swfdisplayitem->setratio** (float ratio)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->setratio() sets the object's ratio to *ratio*. Obviously only useful for morphs.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.

Beispiel 1. swfdisplayitem->setname() example

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
```

```
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor (unknown)

Adds the given color to this item's color transform.

void **swfdisplayitem->addcolor** ([int red [, int green [, int blue [, int a]]]])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->multColor (unknown)

Multiplies the item's color transform.

void **swfdisplayitem->multcolor** ([int red [, int green [, int blue [, int a]]]])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmospher to Halloween (use a landscape or bright picture).

Beispiel 1. swfdisplayitem->multcolor() example

```
<?php

  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape (PHP 4 >= 4.0.5)

Creates a new shape object.

new **swfshape** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** and **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.

Beispiel 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLine (unknown)

Sets the shape's line style.

void **swfshape->setline** (int width [, int red [, int green [, int blue [, int a]]]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->setline() sets the shape's line style. `width` is the line's width. If `width` is 0, the line's style is removed (then, all other arguments are ignored). If `width` > 0, then line's color is set to `red`, `green`, `blue`. Last parameter `a` is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Beispiel 1. swfshape->setline() example

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);
```

```
  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->addFill (unknown)

Adds a solid fill to the shape.

void **swfshape->addfill** (int red, int green, int blue [, int a])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

void **swfshape->addfill** (SWFbitmap bitmap [, int flags])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

void **swfshape->addfill** (SWFGradient gradient [, int flags])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->addfill() adds a solid fill to the shape's list of fill styles. **swfshape->addfill()** accepts three different types of arguments.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

The *bitmap* argument is an swfbitmap() object. The *flags* argument can be one of the following values : SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP. I think.

The *gradient* argument is an swfgradient() object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

swfshape->addfill() returns an swffill() object for use with the **swfshape->setleftfill()** and **swfshape->setrightfill()** functions described below.

See also **swfshape->setleftfill()** and **swfshape->setrightfill()**.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Beispiel 1. swfshape->addfill() example

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap("alphafill.jpg");
  // use your own bitmap
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
```

```
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLeftFill (unknown)

Sets left rasterizing color.

void **swfshape->setleftfill** (swfgradient fill)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

void **swfshape->setleftfill** (int red, int green, int blue [, int a])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

See also **swfshape->setrightfill()**.

SWFShape->setRightFill (unknown)

Sets right rasterizing color.

void **swfshape->setrightfill** (swfgradient fill)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

void **swfshape->setrightfill** (int red, int green, int blue [, int a])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

See also **swfshape->setleftfill()**.

Shortcut for `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFShape->movePenTo (unknown)

Moves the shape's pen.

void **swfshape->movepento** (int x, int y)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->setrightfill() move the shape's pen to (x,y) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->movePen (unknown)

Moves the shape's pen (relative).

void **swfshape->movepen** (int dx, int dy)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->drawLineTo (unknown)

Draws a line.

void **swfshape->drawlineto** (int x, int y)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawline()**.

SWFShape->drawLine (unknown)

Draws a line (relative).

void **swfshape->drawline** (int dx, int dy)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawlineto()**.

SWFShape->drawCurveTo (unknown)

Draws a curve.

void **swfshape->drawcurveto** (int controlx, int controly, int anchorx, int anchory)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (`anchorx`,`anchory`) using (`controlx`,`controly`) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawCurve (unknown)

Draws a curve (relative).

void **swfshape->drawcurve** (int controldx, int controldy, int anchordx, int anchordy)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (`anchorx`,`anchory`) using relative control point (`controlx`,`controly`). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFGradient (PHP 4 >= 4.0.5)

Creates a gradient object

new **swfgradient** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfgradient() creates a new SWFGradient object.

After you’ve added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

Beispiel 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
```

```
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFGradient->addEntry (unknown)

Adds an entry to the gradient list.

void **swfgradient->addentry** (float ratio, int red, int green, int blue [, int a])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfgradient->addentry() adds an entry to the gradient list. `ratio` is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

`red`, `green`, `blue` is a color (RGB mode). Last parameter `a` is optional.

SWFBitmap (PHP 4 >= 4.0.5)

Loads Bitmap object

new **swfbitmap** (string filename [, int alphafilename])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file named `filename`. `alphafilename` indicates a MSK file to be used as an alpha mask for a Jpeg image.

> **Anmerkung:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

Beispiel 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

And you can put an alpha mask on a jpeg fill.

Beispiel 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
```

```
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // draw checkerboard background
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFBitmap->getWidth (unknown)

Returns the bitmap's width.

int **swfbitmap->getwidth** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFBitmap->getHeight (unknown)

Returns the bitmap's height.

int **swfbitmap->getheight** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFFill (PHP 4 >= 4.0.5)

Loads SWFFill object

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods : **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

SWFFill->moveTo (unknown)

Moves fill origin

void **swffill->moveto** (int x, int y)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swffill->moveto() moves fill's origin to (x,y) in global coordinates.

SWFFill->scaleTo (unknown)

Sets fill's scale

void **swffill->scaleto** (int x, int y)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->rotateTo (unknown)

Sets fill's rotation

void **swffill->rotateto** (float degrees)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->skewXTo (unknown)

Sets fill x-skew

void **swffill->skewxto** (float x)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo (unknown)

Sets fill y-skew

void **swffill->skewyto** (float y)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swffill->skewyto() sets fill y-skew to *y*. For *y* is 1.0, it is a is a 45-degree upward slant. More is more upward, less is more downward.

SWFMorph (PHP 4 >= 4.0.5)

Creates a new SWFMorph object.

new **swfmorph** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.

Beispiel 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
```

```
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1 (unknown)

Gets a handle to the starting shape

mixed **swfmorph->getshape1** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an swfshape() object.

SWFMorph->getshape2 (unknown)

Gets a handle to the ending shape

mixed **swfmorph->getshape2** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an swfshape() object.

SWFText (PHP 4 >= 4.0.5)

Creates a new SWFText object.

new **swftext** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Beispiel 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);
```

```
  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->setFont (unknown)

Sets the current font

void **swftext->setfont** (string font)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->setfont() sets the current font to `font`.

SWFText->setHeight (unknown)

Sets the current font height

void **swftext->setheight** (int height)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->setheight() sets the current font height to `height`. Default is 240.

SWFText->setSpacing (unknown)

Sets the current font spacing

void **swftext->setspacing** (float spacing)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->setspacing() sets the current font spacing to `spacingspacing`. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFText->setColor (unknown)

Sets the current font color

void **swftext->setcolor** (int red, int green, int blue [, int a])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->moveTo (unknown)

Moves the pen

void **swftext->moveto** (int x, int y)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*,*y*) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->addString (unknown)

Draws a string

void **swftext->addstring** (string string)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->addstring() draws the string `string` at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth (unknown)

Computes string's width

void **swftext->addstring** (string string)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftext->addstring() returns the rendered width of the `string` string at the text object's current font, scale, and spacing settings.

SWFFont (PHP 4 >= 4.0.5)

Loads a font definition

new **swffont** (string filename)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

If `filename` is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with `<font name="sans-serif">` in HTML.

swffont() returns a reference to the font definition, for use in the **SWFText->setFont()** and the **SWFTextField->setFont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

swffont->getwidth (unknown)

Returns the string's width

int **swffont->getwidth** (string string)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swffont->getwidth() returns the string *`string`*'s width, using font's default scaling. You'll probably want to use the **SWFText()** version of this method which uses the text object's scale.

SWFTextField (PHP 4 >= 4.0.5)

Creates a text field object

new **swftextfield** ([int flags])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield() creates a new text field object. Text Fields are less flexible than swftext() objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_WORDWRAP allows text to wrap
- SWFTEXTFIELD_NOSELECT makes the field non-selectable

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

SWFTextField->setFont (unknown)

Sets the text field font

void **swftextfield->setfont** (string font)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setfont() sets the text field font to the [browser-defined?] `font` font.

SWFTextField->setbounds (unknown)

Sets the text field width and height

void **swftextfield->setbounds** (int width, int height)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setbounds() sets the text field width to `width` and height to `height`. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->align (unknown)

Sets the text field alignment

void **swftextfield->align** (int alignement)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->align() sets the text field alignment to `alignement`. Valid values for `alignement` are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setHeight (unknown)

Sets the font height of this text field font.

void **swftextfield->setheight** (int height)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setheight() sets the font height of this text field font to the given height `height`. Default is 240.

SWFTextField->setLeftMargin (unknown)

Sets the left margin width of the text field.

void **swftextfield->setleftmargin** (int width)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setleftmargin() sets the left margin width of the text field to `width`. Default is 0.

SWFTextField->setrightMargin (unknown)

Sets the right margin width of the text field.

void **swftextfield->setrightmargin** (int width)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setrightmargin() sets the right margin width of the text field to `width`. Default is 0.

SWFTextField->setMargins (unknown)

Sets the margins width of the text field.

void **swftextfield->setmargins** (int left, int right)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setindentation (unknown)

Sets the indentation of the first line.

void **swftextfield->setindentation** (int width)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setindentation() sets the indentation of the first line in the text field, to `width`.

SWFTextField->setLineSpacing (unknown)

Sets the line spacing of the text field.

void **swftextfield->setlinespacing** (int height)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setlinespacing() sets the line spacing of the text field to the height of `height`. Default is 40.

SWFTextField->setcolor (unknown)

Sets the color of the text field.

void **swftextfield->setcolor** (int red, int green, int blue [, int a])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setname (unknown)

Sets the variable name

void **swftextfield->setname** (string name)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setname() sets the variable name of this text field to `name`, for form posting and action scripting purposes.

SWFTextField->addstring (unknown)

Concatenates the given string to the text field

void **swftextfield->addstring** (string string)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swftextfield->setname() concatenates the string `string` to the text field.

SWFSprite (PHP 4 >= 4.0.5)

Creates a movie clip (a sprite)

new **swfsprite** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** and **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Beispiel 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500,1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFSprite->add (unknown)

Adds an object to a sprite

void **swfsprite->add** (resource object)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfsprite->add() adds a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object.

For displayable types (swfshape(), swfbutton(), swftext(), swfaction() or swfsprite()), this returns a handle to the object in a display list.

SWFSprite->remove (unknown)

Removes an object to a sprite

void **swfsprite->remove** (ressource object)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfsprite->remove() remove a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object from the sprite.

SWFSprite->setframes (unknown)

Sets the total number of frames in the animation.

void **swfsprite->setframes** (int numberofframes)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfsprite->setframes() sets the total number of frames in the animation to `numberofframes`.

SWFSprite->nextframe (unknown)

Moves to the next frame of the animation.

void **swfsprite->nextframe** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfsprite->setframes() moves to the next frame of the animation.

SWFbutton (PHP 4 >= 4.0.5)

Creates a new Button.

new **swfbutton** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover() swfbutton->setdown()**, **swfbutton->sethit() swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Beispiel 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200,200);
    return $t;
  }
```

```
function addLabel($string)
{
  global $p;

  $i = $p->add(label($string));
  $p->nextFrame();
  $p->remove($i);
}

$p->add(new SWFAction("stop();"));
addLabel("NO ACTION");
addLabel("SWFBUTTON_MOUSEUP");
addLabel("SWFBUTTON_MOUSEDOWN");
addLabel("SWFBUTTON_MOUSEOVER");
addLabel("SWFBUTTON_MOUSEOUT");
addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
addLabel("SWFBUTTON_DRAGOVER");
addLabel("SWFBUTTON_DRAGOUT");

function rect($r, $g, $b)
{
  $s = new SWFShape();
  $s->setRightFill($s->addFill($r, $g, $b));
  $s->drawLine(600,0);
  $s->drawLine(0,600);
  $s->drawLine(-600,0);
  $s->drawLine(0,-600);

  return $s;
}

$b = new SWFButton();
$b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
$b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
$b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
          SWFBUTTON_MOUSEUP);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
      SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
      SWFBUTTON_MOUSEOVER);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
      SWFBUTTON_MOUSEOUT);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
      SWFBUTTON_MOUSEUPOUTSIDE);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
      SWFBUTTON_DRAGOVER);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
      SWFBUTTON_DRAGOUT);
```

```
  $m = new SWFMovie();
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Beispiel 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("stopDrag();"),
        SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFbutton->addShape (unknown)

Adds a shape to a button

void **swfbutton->addshape** (ressource shape, int flags)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->addshape() adds the shape `shape` to this button. The following `flags'` values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT. SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setUp (unknown)

Alias for addShape(shape, SWFBUTTON_UP)

void **swfbutton->setup** (ressource shape)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setOver (unknown)

Alias for addShape(shape, SWFBUTTON_OVER)

void **swfbutton->setover** (ressource shape)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setdown (unknown)

Alias for addShape(shape, SWFBUTTON_DOWN))

void **swfbutton->setdown** (ressource shape)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setHit (unknown)

Alias for addShape(shape, SWFBUTTON_HIT)

void **swfbutton->sethit** (ressource shape)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->addAction (unknown)

Adds an action

void **swfbutton->addaction** (ressource action, int flags)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->addaction() adds the action `action` to this button for the given conditions. The following `flags` are valid: SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT and SWFBUTTON_DRAGOVER.

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setAction (unknown)

Sets the action

void **swfbutton->setaction** (ressource action)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). `action` is a swfaction().

See also **swfbutton->addshape()** and **SWFAction()**.

SWFAction (PHP 4 >= 4.0.5)

Creates a new Action.

new **swfaction** (string script)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range 0-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (`FALSE`) or 1 (`TRUE`). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given URL into the named target. The target argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional method argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)

Load the given URL into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as `/box.x = 100;`. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a unix filesystem- here it shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Beispiel 1. swfaction() example

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Beispiel 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);
```

```
/* mouse tracking sprite - empty, but follows mouse so we can
   get its x and y coordinates */

$i = $m->add(new SWFSprite());
$i->setName('mouse');

$m->add(new SWFAction("
  startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
"));


/* might as well turn off antialiasing, since these are just squares. */

$m->add(new SWFAction("
  this.quality = 0;
"));

/* morphing box */
$r = new SWFMorph();
$s = $r->getShape1();

/* Note this is backwards from normal shapes.  No idea why. */
$s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
$s->movePenTo(-40, -40);
$s->drawLine(80, 0);
$s->drawLine(0, 80);
$s->drawLine(-80, 0);
$s->drawLine(0, -80);

$s = $r->getShape2();

$s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
$s->movePenTo(-1, -1);
$s->drawLine(2, 0);
$s->drawLine(0, 2);
$s->drawLine(-2, 0);
$s->drawLine(0, -2);


/* sprite container for morphing box -
   this is just a timeline w/ the box morphing */

$box = new SWFSprite();
$box->add(new SWFAction("
  stop();
"));
$i = $box->add($r);

for($n=0; $n<=20; ++$n)
{
  $i->setRatio($n/20);
  $box->nextFrame();
}


/* this container sprite allows us to use the same action code many times */
```

```
  $cell = new SWFSprite();
  $i = $cell->add($box);
  $i->setName('box');

  $cell->add(new SWFAction("

    setTarget('box');

    /* ...x means the x coordinate of the parent, i.e. (..).x */
    dx = (/mouse.x + random(6)-3 - ...x)/5;
    dy = (/mouse.y + random(6)-3 - ...y)/5;
    gotoFrame(int(dx*dx + dy*dy));

  "));

  $cell->nextFrame();
  $cell->add(new SWFAction("

    gotoFrame(0);
    play();

  "));

  $cell->nextFrame();


  /* finally, add a bunch of the cells to the movie */

  for($x=0; $x<12; ++$x)
  {
    for($y=0; $y<8; ++$y)
    {
      $i = $m->add($cell);
      $i->moveTo(100*$x+50, 100*$y+50);
    }
  }

  $m->nextFrame();

  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Same as above, but with nice colored balls...

Beispiel 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
```

```
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if(num == 11)
  num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);
```

```
  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will handles keyboard actions. (You'll probably have to click in the window to give it focus. And you'll probably have to leave your mouse in the frame, too. If you know how to give buttons focus programatically, feel free to share, won't you?)

Beispiel 4. swfaction() example

```
<?php

  /* sprite has one letter per frame */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
       "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&/*()_+-=/[]{}|;:,.<>¿~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }


  /* hit region for button - the entire frame */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);


  /* button checks for pressed key, sends sprite to the right frame */
```

```
  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }

  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

LVIII. Sonstige Funktionen

Diese Funktionen stehen hier, weil sie in keine andere Kategorie passen.

connection_aborted (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ergibt `TRUE`, wenn die Client-Verbindung beendet wurde

int **connection_aborted** (void)

Gibt `TRUE` zurück, wenn der Client nicht mehr verbunden ist. Siehe auch die Connection Handling-Beschreibung im Features-Kapitel für weitere Erläuterungen.

connection_status (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ergibt den Verbindungs-Status als Bit-Feld

int **connection_status** (void)

Gibt den Verbindungs-Status als Bit-Feld zurück.Siehe auch die Connection Handling-Beschreibung im Features-Kapitel für weitere Erläuterungen.

connection_timeout (PHP 3>= 3.0.7, 4.0.0 - 4.0.4 only)

Ergibt `TRUE`, wenn das Skript seinen Time-Out erreicht hat

int **connection_timeout** (void)

Gibt `TRUE` zurück, wenn der Time-Out des Skripts erreicht ist. Siehe auch die Connection Handling-Beschreibung im Features-Kapitel für weitere Erläuterungen.

define (PHP 3, PHP 4 >= 4.0.0)

Definiert eine Konstante

int **define** (string name, mixed value [, int case_insensitive])

Definiert eine Konstante, ähnlich wie bei Variablen, ausser das:

- Eine Konstante nicht mit einem '$' beginnt;
- Auf eine Konstante überall zugegriffen werden kann, ohne allerdings den Auswertungsregeln von Variablen zu unterliegen;
- Eine Konstante nicht umdefiniert oder zurück gesetzt werden kann, wenn sie einmal gesetzt wurde und
- eine Konstante nur eindimensionale Werte (also keine Arrays) enthalten kann.

Der Konstanten-Name wird durch *`name`* angegeben. Der Wert wird durch *`value`* gesetzt.

Wenn der optional mögliche dritte Parameter *`case_insensitive`* den Wert *1* hat, wird die Konstante zwischen Groß- und Kleinschreibung unterschieden. Das ist auch die Voreinstellung (es wird dann also zwischen KONSTANTE und Konstante unterschieden).

Die Funktionalität des dritten Parameters scheint z.Zt. noch nicht richtig umgesetzt worden zu sein (Stand 03/2000).

Beispiel 1. Definition von Konstanten:

```
<?php
define ("KONSTANTE", "Hallo Welt.");
echo KONSTANTE; // outputs "Hallo Welt.";
?>
```

define() gibt bei erfolgreicher Zuweisung `TRUE`, sonst `FALSE` zurück.

Siehe auch unter defined() und im Abschnitt Konstanten.

defined (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine angegebene Konstante existiert

int **defined** (string name)

Wenn die angegebene Konstante *`name`* existiert, wird `TRUE` zurück gegeben, ansonsten `FALSE`.

Siehe auch define() und den Abschnitt Konstanten.

die (unknown)

Gibt eine Nachricht aus und beendet das aktuelle Skript

void **die** (string message)

Dieser Sprach-Konstrukt erzeugt eine Meldung und bricht das Parsing / die Abarbeitung des Skriptes ab (ohne Rückkehrmöglichkeit).

Beispiel 1. "Die"-Beispiel:

```
<?php
$filename = '/pfad/zu/dateiname';
$file = fopen ($filename, 'r')
    or die ("Kann die Datei ($filename) nicht öffnen.");
?>
```

Beachten sie auch die Funktion exit().

eval (unknown)

Wertet einen String aus, als wäre er PHP-Code

mixed **eval** (string code_str)

eval() versucht, den in `code_str` enthaltenen String als PHP-Code auszuführen. Abgesehen von anderen Zwecken kann dies nützlich sein, um Code auszuführen, der aus einer Datenbank ausgelesen wird.

Beim Gebrauch von **eval()** gilt es einige Faktoren zu beachten. Die Zeichenkette `code_str` muss gültigen PHP-Code enthalten, einschließlich syntaktisch richtiger Begrenzungszeichen (also incl. ";"-Abschluss-Zeichen für einzelne Anweisungen, sonst bricht der PHP-Parser in der folgenden Zeile mit einem Fehler ab.

Beachten sie, dass die innerhalb von **eval()** vorgenommenen Variablen-Zuweisungen nach Abarbeitung auch im Hauptskript ihre Gültigkeit behalten.

Eine `return`-Angabe beendet sofort die Auswertung des Strings. Mit PHP4 können sie ein `return` verwenden, um das Ergebnis der Funktion **eval()** zu erhalten. In PHP3 ist **eval()** nämlich vom Typ `void` und liefert nichts zurück.

Beispiel 1. eval()-Beispiel (einfache Texteinfügung):

```
<?php
$string = 'Bierglas';
$name = 'Binding-Lager';
$str = 'Das ist mein $string, voll mit $name.<br>';
echo $str;
eval ("\$str = \"$str\";");
echo $str;
?>
```

Das obige Beispiel gibt aus:

```
Das ist mein $string, voll mit $name.
Das ist mein Bierglas, voll mit Binding-Lager.
```

Beachten sie bitte das Setzen des Begrenzungs-Zeichens ";" in `code_str` sowie den Gebrauch des "\"-Zeichens.

exit (unknown)

Beenden des aktuellen Skripts

void **exit** (void)

Mit **exit()** beenden sie - ohne Möglichkeit der Rückkehr - die Skriptausführung.

Beachten sie auch die Funktion die().

get_browser (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die Möglichkeiten des Browsers eines Benutzers

object **get_browser** ([string user_agent])

get_browser() ermittelt die Möglichkeiten des auf das PHP-Skript zugreifenden Browsers. Dies geschieht durch Auslesen der zum Browser vorhandenen Werte aus der Datei `browscap.ini`. Per Voreinstellung wird der Wert von $HTTP_USER_AGENT als Browser-Typ benutzt. Dieses Verhalten können sie ändern (so z.B. den Namen der INI-Datei), indem sie den optionalen *`user_agent`*-Parameter an die **get_browser()**-Funktion übergeben.

Die zurück gegebene Information ist ein Objekt, das verschiedene Datenelemente enthält. Dazu gehören z.B. die Haupt- und Unterversions-Nummer des Browsers sowie `TRUE` / false-Werte für Browser-*Fähigkeiten* (u.a. Frame-, Java- und Cookies-Unterstützung). Nicht gemeint ist hier die aktuell vorhandene Nutzung. Ist der Browser so konfiguriert, dass er z.Zt. keine Cookies annehmen soll, wird trotzdem `TRUE` in Bezug auf die Cookie-Fähigkeiten zurückgegeben, wenn in der Datei `browscap.ini` der entsprechende Wert `TRUE` ist.

Da in `browscap.ini` Informationen vieler Browser stehen, obliegt es dem Skript-Schreiber, diese Datei auf dem Laufenden zu halten. Das Format dieser Datei sollte selbsterklärend sein.

Das folgende Beispiel zeigt, wie man alle verfügbaren Informationen zu einem gerade benutzten Browser anzeigen könnte:

Beispiel 1. get_browser()-Beispiel:

```
<?php
function list_array ($array) {
    while (list ($key, $value) = each ($array)) {
        $str .= "<b>$key:</b> $value<br>\n";
    }
    return $str;
}
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array ((array) $browser);
?>
```

Die Ausgabe des obigen Skripts könnte so aussehen:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr>
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br>
<b>parent:</b> Netscape 4.0<br>
<b>platform:</b> Unknown<br>
<b>majorver:</b> 4<br>
<b>minorver:</b> 5<br>
<b>browser:</b> Netscape<br>
<b>version:</b> 4<br>
<b>frames:</b> 1<br>
<b>tables:</b> 1<br>
<b>cookies:</b> 1<br>
<b>backgroundsounds:</b> <br>
<b>vbscript:</b> <br>
```

```
<b>javascript:</b> 1<br>
<b>javaapplets:</b> 1<br>
<b>activexcontrols:</b> <br>
<b>beta:</b> <br>
<b>crawler:</b> <br>
<b>authenticodeupdate:</b> <br>
<b>msn:</b> <br>
```

Damit dieses Skript richtig funktioniert, muss die browscap-Einstellung ihrer Konfigurations-Datei auf die richtige `browscap.ini`-Datei zeigen.

Weitere Informationen (incl. Bezugsquellen für `browscap.ini`-Dateien) finden sie in der PHP-FAQ unter http://www.php.net/FAQ.php.

highlight_string (PHP 4 >= 4.0.0)

Hervorhebung der Syntax eines Strings

void **highlight_string** (string str)

Die Funktion **highlight_string()** gibt den String *`str`* mit hervorgehobener Syntax (Syntax highlighting) aus. Dabei werden die Farben des in PHP eingebauten Syntax-Highlighter benutzt.

Siehe auch highlight_file() und show_source().

highlight_file (PHP 4 >= 4.0.0)

Syntax-Hervorhebung für eine Datei

void **highlight_file** (string filename)

Die Funktion **highlight_file()** erzeugt die Ausgabe des Codes der Datei *`filename`* mit hervorgehobener Syntax. Dabei werden die Farben des in PHP eingebauten Syntax-Highlighter benutzt.

Beispiel 1. Erzeugung einer hervorgehobenen URL

Zur Erzeugung einer URL, die Code eines ihm übermittelten Skripts hervorgehoben dargestellen kann, machen wir Gebrauch von der "ForceType"-Anweisung des Apachen um eine richtige URL zu erzeugen. Zusätzlich benutzen wir die Funktion **highlight_file()**, um das Code-Listing wie gewünscht an zu zeigen.

In ihrer httpd.conf sollten sie folgendes hinzu fügen:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

Jetzt erzeugen sie eine Datei namens "source" und speichern es in dem Root-Verzeichnis ihrer Web-Seite.

```
<HTML>
<HEAD>
<TITLE>Source Display</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
        echo "<BR><B>Fehler: Skript-Name nötig</B><BR>";
    } else {
        if (ereg("(\.php|\.inc)$",$script)) {
            echo "<H1>Source von: $PATH_INFO</H1>\n<HR>\n";
            highlight_file($script);
        } else {
            echo "<H1>ERROR: Nur PHP oder Include-Skripte sind erlaubt</H1>";
        }
    }
    echo "<HR>Erzeugt: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Jetzt können sie die u.a. URL benutzen, um eine farblich hervorgehobene Version des Skriptes "/path/to/script.php" innerhalb ihrer Web-Seite zu zeigen.

```
http://ihr.server.de/source/pfad/zu/script.php
```

Siehe auch highlight_string() und show_source().

ignore_user_abort (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Stellt ein, ob der Verbindungsabbruch eines Clients die Skript-Ausführung abbrechen soll

int **ignore_user_abort** ([int setting])

Mit dieser Funktion wird der Wert dafür gesetzt, ob der Abbruch einer Client-Verbindung die weitere Abarbeitung eines Skripts beenden soll. Es wird der vorherige Wert zurück gegeben. Die Funktion kann auch ohne Argument aufgerufen werden, um nur die aktuelle Einstellung zu erfahren. Die komplette Beschreibung des Verbindungs-Handling von PHP können sie im Abschnitt Connection-Handling des Kapitels Features nachlesen. Dort werden auch die Werte für den optionalen Parameter *setting* näher erläutert.

iptcparse (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Übersetzt einen binären IPTC-http://www.iptc.org/-Block in einzelne Tags.

array **iptcparse** (string iptcblock)

Mit dieser Funktion übertragen sie einen binären IPTC-Block in seine einzelnen Tags. Es wird ein Array zurück gegeben. In diesem bilden die Tag-Markierungen den Index, die Werte stellen den Inhalt dar. Bei einem Fehler oder wenn keine IPTC-Daten gefunden wurden, wird FALSE zurück gegeben. Unter **GetImageSize()** finden sie ein Beispiel.

leak (PHP 3, PHP 4 >= 4.0.0)

Speicher-Verbrauch

void **leak** (int bytes)

leak() belegt die angegebene Speichergrösse.

Diese Funktion ist hilfreich beim Debuggen des Speicher-Managers, der "leaked"-Speicher nach Abarbeitung aller Anforderungen wieder automatisch frei gibt.

pack (PHP 3, PHP 4 >= 4.0.0)

Packt Daten in einen Binär-String

string **pack** (string format [, mixed args])

Diese Funktion packt ein oder mehrere Argumente - unter Beachtung der Argument-Formate - in einen Binär-String. In dem zurück gegebenen Binär-String stehen anschließend die Daten.

Die Idee entstammt Perl. Alle Formatierungs-Anweisungen funktionieren genau wie dort. In PHP fehlen aber einige Format-Codes von Perl (z.B. "u"). Der Format-String besteht aus Format-Codes, gefolgt von einem optionalen Wiederholungs-Argument. Dieses Argument kann ein Integer-Wert oder ein "*" (für Wiederholung bis ans Ende der Daten) sein. Bei "a", "A", "h" und dem "H" Format-Code gibt das Wiederholungs-Argument an, wie viele gleiche Zeichen folgen. Im Zusammenhang mit "@" gibt das Wiederholungs-Argument die absolute Position an, ab der das nächste Zeichen steht. Bei allen anderen steht der Wiederholungs-Zähler für die Anzahl der benutzten Daten-Argumente, die in den Binär-String gepackt werden sollen. Z. Zt. sind folgende Codes implementiert:

- a mit NUL gefüllte Zeichenkette (String)
- A mit Leerzeichen (SPACE) gefüllte Zeichenkette (String)
- h Hex Zeichenkette (String), unterer Halbwert zuerst
- H Hex Zeichenkette (String), höherer Halbwert zuerst
- c signed char
- C unsigned char
- s Short-Typ mit Vorzeichen (immer 16 Bit, Byte-Folge maschinenabhängig)

- S Short-Typ ohne Vorzeichen (immer 16 Bit, Byte-Folge maschinenabhängig)
- n Short-Typ ohne Vorzeichen (immer 16 Bit, big endian byte order)
- v Short-Typ ohne Vorzeichen (immer 16 Bit, little endian byte order)
- i Integer mit Vorzeichen (Größe und Byte-Folge maschinenabhängig)
- I Integer-Typ ohne Vorzeichen (Größe und Byte-Folge maschinenabhängig)
- l Long-Typ mit Vorzeichen (immer 32 Bit, Byte-Folge maschinenabhängig)
- L Long-Typ ohne Vorzeichen (immer 32 Bit, Byte-Folge maschinenabhängig)
- N Long-Typ ohne Vorzeichen (immer 32 Bit, big endian Byte- Folge)
- V Long-Typ ohne Vorzeichen (immer 32 Bit, little endian Byte- Folge)
- f Fließkomma-Typ (maschinenabhängige Größe und Wiedergabe)
- d Double-Typ (maschinenabhängige Größe und Wiedergabe)
- x NUL Byte
- X Sichern eines Bytes
- @ NUL-Auffüllung bis zur absoluten Position

Beispiel 1. Mit Pack formatierter String:

```
$binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
```

Der sich daraus ergebende Binär-String ist 6 Bytes lang und enthält: 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Bedenken sie, dass der Unterschied zwischen vorzeichen-losen und vorzeichen-behafteten Werten nur Einfluss auf die Funktion unpack() hat, wogegen dies bei der Funktion **pack()** egal ist.

Beachten sie auch, dass PHP intern Ganzzahl-Werte als vorzeichen-behaftete Werte speichert, deren Umfang vom Maschinen-Typ abhängig ist. Geben sie PHP einen vorzeichen-losen Ganzzahl-Wert, der zu groß ist für diese Art der Speicherung, wird er ins Double-Format umgesetzt, was oft zu unerwünschten Resultaten führt.

show_source (PHP 4 >= 4.0.0)

Syntax-Highlighting einer Datei

void **show_source** (string filename)

Die Funktion **show_source()** erzeugt die Ausgabe des Codes der Datei *`filename`* mit hervorgehobener Syntax. Dabei werden die Farben des in PHP eingebauten Syntax-Highlighter benutzt.

Anmerkung: Diese Funktion ist ein Alias der Funktion highlight_file().

Siehe auch highlight_string() und highlight_file().

sleep (PHP 3, PHP 4 >= 4.0.0)

Programmverzögerung

void **sleep** (int seconds)

Die **sleep()**-Funktion unterbricht die weitere Skript-Ausführung für die mit `seconds` angegebene Anzahl in Sekunden.

Siehe auch usleep().

uniqid (PHP 3, PHP 4 >= 4.0.0)

Erzeugt eine eindeutige ID

int **uniqid** (string prefix [, boolean lcg])

uniqid() gibt eine eindeutige ID mit Präfix zurück. Diese basiert auf der aktuellen Zeit in Mikrosekunden. Das Präfix kann hilfreich sein, wenn sie z. B. auf mehreren Servern gleichzeitig ID´s erzeugen, was ja im ungünstigsten Fall zur selben Mikrosekunde geschehen kann. Der `prefix`-Parameter kann bis zu 114 Zeichen lang sein.

Ist der optionale Parameter `lcg` `TRUE`, wird **uniqid()** am Ende des Rückgabewertes zusätzliche "LCG"-Zustandswerte anfügen, durch die das Ergebnis noch eindeutiger wird.

Bei leerem `prefix`-Parameter ist der zurück gegebene String 13 Zeichen lang. Ist der Parameter `lcg` `TRUE`, sind es 23 Zeichen.

Anmerkung: Der `lcg`-Parameter ist erst ab PHP 4 bzw. PHP 3.0.13 und später verfügbar.

Wenn sie eindeutige ID´s bzw. eindeutige Token benötigen und beabsichtigen, diese Token über das Netzwerk an den Benutzer weiter zu leiten (z.B. als Session-Cookies), ist es ratsam, wenn sie wie folgt vorgehen:

```
$token = md5 (uniqid ("")); // kein Zufalls-Anteil = schlecht
$better_token = md5 (uniqid (rand())); // besser, da schwer zu erraten
```

Hierdurch wird eine 32-Zeichen-ID (128 Bit-Hex-Wert) erzeugt, die nur sehr schwer vorhersehbar ist.

unpack (PHP 3, PHP 4 >= 4.0.0)

Entpackt die Daten eines Binär-Strings

array **unpack** (string format, string data)

unpack() überträgt die Daten eines Binär-Strings in ein Array unter Berücksichtigung des *`format`*-Parameters. Das Array wird als Ergebnis des Funktionsaufrufs zurück gegeben.

unpack() funktioniert etwas anders als man es von Perl her kennt, da sich die zurück gegebenen Daten in einem assoziativen Array befinden. Dies erreichen sie nur, wenn sie die verschiedenen Format-Codes benennen und sie mit einem "/" voneinander trennen.

Beispiel 1. unpack() formatierter String

```
$array = unpack ("c2chars/nint", $binarydata);
```

Das resultierende Array wird die Einräge "chars1", "chars2" und "int" enthalten.

Die Format-Codes werden unter pack() näher erläutert.

Beachten sie, dass PHP Integer-Werte intern mit Vorzeichen speichert. Wenn sie einen grossen vorzeichen-losen Long-Wert entpacken und er von der selben Grösse ist, die PHP für dessen Speicherung verwendet, wird das Ergebnis ein negativer Wert sein (auch wenn sie dieses als vorzeichen-los zu entpacken angegeben haben).

usleep (PHP 3, PHP 4 >= 4.0.0)

Programm-Verzögerung in Mikrosekunden

void **usleep** (int micro_seconds)

Mittels der Funktion **usleep()** verzögern sie die weitere Skript-Ausführung um die gegebene Anzahl an *`micro_seconds`*.

Beachten sie auch die Funktion sleep().

Anmerkung: Dies Funktion steht unter Windows-Systemen nicht zur Verfügung.

LIX. mnoGoSearch Funktionen

Diese Funktionen erlauben den Zugriff auf die mnoGoSearch früher als UdmSearch bekannt) Suchmaschine. Um auf diese Funktionen zugreifen zu können, muß php mittels der `--with-mnogosearch`-Option mit mnoGoSearch-Unterstützung compiliert worden sein. Wird diese Option ohne Pfadangabe zu mnoGoSearch verwendet, wird mnoGoSearch in /usr/local/mnoGoSearch gesucht. Wurde mnoGoSearch unter einem anderen Pfad installiert, kann der Pfad mittels `--with-mnogosearch=DIR` spezifiziert werden.

mnoGoSearch ist eine vollwertige Suchmaschine für den Intra- und Interneteinsatz, vertrieben unter der GPL. mnoGoSearch hat einige einzigartige Funktionen, die es für weite Einsatzbereiche tauglich machen. Es bietet Volltextindizierung von HTML, PDF und Textdokumenten. mnoGoSearch besteht aus zwei Teilen: Der erste ist der sogenannte indexer, der den Indizierungsmechanismus bereitstellt. Er spidered HTTP, FTP und Newsserver oder lokale Dateisysteme, indiziert die gefundenen Dokumente und speichert Meta-Daten über die Dokumente in einer SQL-Datenbank.

> **Anmerkung:** php beinhaltet zwar einen eingebauten MySQL-Support, dieser ist jedoch nicht mit mnoGoSearch kompatibel, deshalb muß php gegen die selben generischen MySQL-Bibliotheken wie mnoGoSearch gelinkt werden, also z.B.: `--with-mnogosearch --with-mysql=/usr`.

Es wird mindestens Version 3.1.10 von mnoGoSearch vorausgesetzt, um die folgenden Funktionen nutzen zu können.

Weitere Informationen über mnoGoSearch können unter http://www.mnogosearch.ru/ abgerufen werden.

udm_add_search_limit (PHP 4 >= 4.0.5)

Verschiedene Beschränkungen bei der Suche anwenden

int **udm_add_search_limit** (int agent, int var, string val)

udm_add_search_limit() gibt `TRUE` bei Erfolg, `FALSE` bei einem Fehler zurück. Fügt Suchrestritktionen hinzu.

agent - ein Link zu einem Agent, erhalten nach udm_alloc_agent().

var - Definiert den Parameter der die Einschränkung angibt.

val - Definiert den Wert des aktuellen Parameters.

Mögliche Werte für *var*:

- UDM_LIMIT_URL - definiert URL Limitationen, um die Suche auf Unterbereiche einzuschränken SQL % und _ LIKE Wildcards werden unterstützt, % entspricht dabei einer beliebigen Anzahl von Zeichen, wobei auch keine Zeichen eingeschlossen werden, _ entspricht genau einem Zeichen. So entspricht also http://my.domain__/catalog sowohl http://my.domain.ru/catalog als auch http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - definiert TAG Limitationen. In der indexer-conf können bestimmeten Sites und Teilen von Sites sogenannte Tags zugewiesen werden. Auch bei Tags werden % und _ unterstützt. Metasymbole erlauben die Suche über Gruppen von Tags. So werden also die Tags ABCD und ABCE bei einer Einschränkung auf ABC_ eingeschlossen.
- UDM_LIMIT_LANG - Definiert Sprachlimitationen.
- UDM_LIMIT_CAT - Definiert Dokumentenkategorielimitationen. Kategorien sind den Tags ähnlich, jedoch im Gegensatz zu diesen verschachtelt. Eine Kategorie kann also Unterkategorie einer anderen sein. Für jede Ebene werden müssen zwei Zeichen verwendendet werden, entweder hexadezimale Zahlen von 0-F oder Zahlen zur Basis 36 von 0-Z. Eine Kategorie der obersten Ebene, z.B. 'Auto' hätte die Zahl 01, die Unterkategorie 'Ford' hätte ebenfalls die 01 als Zahl. Zusammen ergäbe das dann die 0101. Eine weitere Kategorie 'VW' unterhalb der Kategorie 'Auto' hätte dann die Zahl 0102 usw. Wenn 'VW' eine Unterkategorie 'Golf' hätte, wäre desren ID 010201.
- UDM_LIMIT_DATE - Definiert Einschränkungen in Bezug auf das Datum, wann das Dokument zuletzt verändert wurde.

  Der Wert dieses Parameters ist das kleiner bzw. größer-Zeichen (< oder >) gefolgt von einem Unix-Timestamp, also z.B.

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  Wenn das >-Zeichen benutzt wird, wird die Suche auf diejenigen Dokumente eingeschränkt, deren Veränderungsdatum jünger als das eingegebene ist, beim <-Zeichen erfolgt die Einschränkung dementsprechend auf Dokumente, die Älter sind.

udm_alloc_agent (PHP 4 >= 4.0.5)

Allocate mnoGoSearch session

int **udm_alloc_agent** (string dbaddr [, string dbmode])

udm_alloc_agent() Gibt bei Erfolg einen eindeutigen Zeiger auf einen mnoGoSearch-Agenten zurück, bei einem Fehler wird `FALSE` zurückgegeben. Diese Funktion kreiert eine Sitzung mit Datenbankparametern.

`dbaddr` - Datenbankadresse im URL-Stil. Wird verwendet, um mit den Daten Datenbanktyp, Datenbankhost, Datenbankname, Datenbankport, Datenbankuser und Datenbankpasswort eine Verbindung zu einer SQL-Datenbank herzustellen. Das Format lautet: DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/.

`dbmode` - Mit dem Parameter dbmode wird der Modus spezifiziert, in dem die Daten in der Datenbank vorliegen. Wird der Modus 'single' gewählt, werden alle Wörter in der selben Tablelle gespeichert. Der Modus 'multi' speichert die Wörter in verschiedenene Tabellen, abhängig von deren Länge. Weitere Modi sind der CRC-Modus und der Multi-CRC-Modus, wobei der CRC-Modus schneller als der Single- und der Multimodus sind, und der Multi-CRC-Modus die Technik des CRC-Modus mit der Speicherstruktur des Multi-Modus verbindet. Format: DBMode single/multi/crc/crc-multi

Anmerkung: *`dbaddr`* und *`dbmode`* müssen mit den Werten übereinstimmen, die beim Indizieren verwendet wurden.

Anmerkung: Eigentlich öffnet diese Funktion keine Verbindung zur Datenbank, folglich werden auch keine User- und Passwortüberprüfungen durchgeführt. Die Verbindung zur Datenbank wird von der Funktion udm_find() übernommen, hier findet dann auch eine Überprüfung der Logindaten statt.

udm_api_version (PHP 4 >= 4.0.5)

Gibt die Version der mnoGoSearch API zurück.

int **udm_api_version** (void)

udm_api_version() gibt die mnoGoSearch API-Versionsnummer zurück. Wird als z.B. die mnoGoSearch 3.1.10 API benutzt, gibt diese Funktion `30110` zurück.

Diese Funktion erlaubt es dem Benutzer zu erfahren, welche Funktionen der mnoGoSearch-API verfügbar sind. udm_get_doc_count() diese Funktion ist erst ab mnoGoSearch 3.1.11 oder später verfügbar.

Beispiel:

```
if (Udm_Api_Version() >= 30111) {
print  "Total number of urls in database: ".Udm_Get_Doc_Count($udm)."<br>\n";
}
```

udm_cat_path (PHP 4 >= 4.0.6)

Liefert den Pfad zur aktuellen Kategorie zurück.

array **udm_cat_path** (int agent, string category)

udm_cat_path() gibt einen Array zurück, der den Pfad im Kategorienbaum von der Wurzel an bis zur aktuellen Kategorie enthält.

agent - Zeiger auf den Agenten.

category - aktuelle Kategorie - diejenige, zu der der Pfad herausgefunden werden soll.

Gibt einen Array mit dem folgenden Format zurück:

Der Array besteht aus Paaren. Elemente mit geraden Indexnummern beinhalten die Kategoriepfade, die ungeraden Elemente behinhalten die entsprechenden Kategorienamen.

So könnte der Aufruf `$array=udm_cat_path($agent, '02031D');` folgenden Array zurückliefern:

$array[0] beinhaltet ”
$array[1] beinhaltet ’Root’
$array[2] beinhaltet ’02’
$array[3] beinhaltet ’Sport’
$array[4] beinhaltet ’0203’
$array[5] beinhaltet ’Auto’
$array[4] beinhaltet ’02031D’
$array[5] beinhaltet ’Ferrari’

Beispiel 1. Spezifiziert den Pfad zu der aktuellen Kategorie in folgendem Format: ’> Root > Sport > Auto > Ferrari’

```
<?php
  $cat_path_arr=Udm_Cat_Path($udm_agent,$cat);
  $cat_path=”;
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path=$cat_path_arr[$i];
    $name=$cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$PHP_SELF?cat=$path\">$name</a> ";
  }
>
```

udm_cat_list (PHP 4 >= 4.0.6)

Liefert alle Kategorien auf der selben Ebene wie der aktuellen zurück.

array **udm_cat_list** (int agent, string category)

udm_cat_list() Gibt einen Array zurück, der alle Kategorien auf dem selben Level im Kategorienbaum wie die aktuelle Kategorie liegen.

Diese Funktion kann nützlich sein, um einen Kategorienbrowser zu entwickeln.

Gibt einen Array mit dem folgenden Format zurück:

Der Array besteht aus Paaren. Elemente mit geraden Indexnummern beinhalten die Kategoriepfade, die ungeraden Elemente behinhalten die entsprechenden Kategorienamen.

$array[0] beinhaltet ’020300’
$array[1] beinhaltet ’Audi’
$array[2] beinhaltet ’020301’
$array[3] beinhaltet ’BMW’
$array[4] beinhaltet ’020302’
$array[5] beinhaltet ’Opel’
...
usw.

Das folgende Beispiel gibt Links im folgenden Format aus:
Audi
BMW
Opel
...

```
<?php
 $cat_list_arr=Udm_Cat_List($udm_agent,$cat);
 $cat_list=”;
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
 $path=$cat_list_arr[$i];
 $name=$cat_list_arr[$i+1];
 $cat_list .= "<a href=\"$PHP_SELF?cat=$path\">$name</a><br>";
 }
>
```

udm_clear_search_limits (PHP 4 >= 4.0.5)

Löscht alle mnoGoSearch Suchrestriktionen

int **udm_clear_search_limits** (int agent)

udm_clear_search_limits() setzt definierte Sucheinschränkungen zurück und gibt `TRUE` zurück.

udm_errno (PHP 4 >= 4.0.5)

Liefert den mnoGoSearch Fehlercode zurück.

int **udm_errno** (int agent)

udm_errno() liefert den mnoGoSearch Fehlercode. Null wenn kein Fehler vorliegt..

`agent` - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

Enthält einen numerischen Fehlercode.

udm_error (PHP 4 >= 4.0.5)

Liefert eine mnoGoSearch Fehlermeldung zurück.

string **udm_error** (int agent)

udm_error() gibt eine mnoGoSearch Fehlermeldung zurück, eine leere Zeichenkette, wenn kein Fehler vorlag.

agent - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

Liefert die Fehlermeldung des Agenten zurück.

udm_find (PHP 4 >= 4.0.5)

Führe eine Suche durch.

int **udm_find** (int agent, string query)

udm_find() gibt einen Ergebniszeiger bei Erfolg zurück, FALSE bei einem Fehler.

Die eigentliche Suche. Das erste Argument - die Session, das nächste - die Suche selbst. Um etwas zu finden, einfach die zu suchenden Begriffe in das Suchfeld eintippen und den Submit-Button drücken. Zum Beispiel "mysql odbc". Die doppelten Anführungszeichen " sollten nicht in einer Suche verwendet werden, sie werden hier nur aufgeführt, um die Suchbegriffe vom anderen Text zu unterscheiden. mnoGoSearch wird alle Dokumente finden, die das Wort "mysql" und/oder "odbc" enthalten. Dokumente, die der Suche besser entsprechen als andere werden zuerst angezeigt. Wird der Suchmodus ALL verwendet, werden nur Dokumente gefunden, die alle gesuchten Wörter mindestens einmal enthalten. Wird der Suchmodus ANY verwendet, werden alle Dokumente zurückgliefert, die eines der gesuchten Wörter enthalten. Wenn ausgefeiltere Suchen stattfinden sollen, muß die Suchsprache verwendet werden. In diesem Falle sollte der boolsche Übereinstimmungsmodus gewählt werden.

mnoGoSearch kennt die folgenden boolschen Operatoren:

& - logisches UND. Zum Beispiel "mysql & odbc". mnoGoSearch wird alle Dokumente finden, die sowohl "mysql" als auch "odbc" enthalten.

| - logisches ODER. Zum Beispiel "mysql|odbc". mnoGoSearch wird alle Dokumente zurückliefern die das Wort "mysql" oder das Wort "odbc" enthalten.

~ - logisches NICHT. Zum Beispiel "mysql & ~odbc". mnoGoSearch wird alle Dokumente finden, die das Wort "mysql" enthalten, nicht aber das Wort "odbc" zur gleichen Zeit. Dieser Operator ist jedoch ein rein ausschließender Operator. Die Suche "~odbc" wird keine Ergebnisse zurückliefern!

() - Gruppierung, um komplexe Suchen zu bilden. Zum Beispiel "(mysql | msql) & ~postgres". Die Suchesprache ist simpel und stark zugleich.

udm_free_agent (PHP 4 >= 4.0.5)

Löscht eine mnoGoSearch Session

int **udm_free_agent** (int agent)

udm_free_agent() gibt TRUE bei Erfolg zurück, FALSE bei einem Fehler.

agent - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

Gibt den Speicher frei, den eine mnoGoSearch-Session belegt hat.

udm_free_ispell_data (PHP 4 >= 4.0.5)

Gibt den Speicher frei, der für ISpell alloziiert wurde.

int **udm_free_ispell_data** (int agent)

udm_free_ispell_data() liefert stets TRUE zurück.

agent - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

Anmerkung: Diese Funktion wird seit Version 3.1.12 von mnoGoSearch unterstützt. In vorherigen Versionen ist sie ohne Funktion.

udm_free_res (PHP 4 >= 4.0.5)

Löscht ein mnoGoSearch-Ergebis.

int **udm_free_res** (int res)

udm_free_res() gibt TRUE bei Erfolg, FALSE bei einem Fehler zurück.

agent - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

Gibt den Speicher frei, der für Ergebnisse alloziiert wurde.

udm_get_doc_count (PHP 4 >= 4.0.5)

Liefert die Anzahl aller Dokumente in der Datenbank zurück.

int **udm_get_doc_count** (int agent)

udm_get_doc_count() gibt die Anzahl der Dokumente in der Datenbank zurück.

agent - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

Anmerkung: Diese Funktion wird seit mnoGoSearch 3.1.11 unterstützt.

udm_get_res_field (PHP 4 >= 4.0.5)

Erhalte ein mnoGoSearch Ergebenisfeld

string **udm_get_res_field** (int res, int row, int field)

udm_get_res_field() gibt den Wert eines Ergebnisfeldes bei Erfolg, `FALSE` bei einem Fehler zurück.

`agent` - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

`row` - Die Zahl des aktuellen Links. Werte können zwischen 0 und *`UDM_PARAM_NUM_ROWS-1`* variieren.

`field` - Feldzeiger, kann folgende Werte annehmen:

- UDM_FIELD_URL - URL des Dokumentes
- UDM_FIELD_CONTENT - Content-Type des Dokumentes (zum Beispiel text/html).
- UDM_FIELD_CATEGORY - Kategorie des Dokumentes. udm_cat_path() sollte verwendet werden, um den vollen Pfad vom der Wurzel des Kategorienbaumes zur aktuellen Kategorie zu erhalten. (Dieser Parameter ist erst ab PHP Version 4.0.6 verfügbar).
- UDM_FIELD_TITLE - Titel des Dokumentes.
- UDM_FIELD_KEYWORDS - Keywords des Dokumentes (aus dem Meta-Tag Keywords).
- UDM_FIELD_DESC - Dokumentenbeschreibung (aus dem Meta-Tag Description).
- UDM_FIELD_TEXT - Inhalt des Dokumentes (die ersten Zeilen des Dokumentes).
- UDM_FIELD_SIZE - Größe des Dokumentes.
- UDM_FIELD_URLID - Eindeutige ID des Dokumentes.
- UDM_FIELD_RATING - Dokumentenwertung (von mnoGoSearch berechnet).
- UDM_FIELD_MODIFIED - Datum der letzten Änderung des Dokumentes als Unix-Timestamp.
- UDM_FIELD_ORDER - Die Nummer des aktuellen Dokumentes in der Menge der gefundenen Dokumente.
- UDM_FIELD_CRC - Die Prüfsumme des Dokumentes.

udm_get_res_param (PHP 4 >= 4.0.5)

Liefert die mnoGoSearch Ergbnisparameter zurück.

string **udm_get_res_param** (int res, int param)

udm_get_res_param() gibt den Wert des Parameters bei Erfolg. `FALSE` bei einem Fehler zurüc.

`res` - ein Ergebniszeiger, erhalten nach einem Aufruf von udm_find().

`param` - Parameterbezeichner. Kann folgenden Werte annehmen:

- UDM_PARAM_NUM_ROWS - Anzahl der auf der aktuellen Seite gefundenen Links.
- UDM_PARAM_FOUND - Anzahl der gefundenen Dokuemnte, die der Suche entsprechen.
- UDM_PARAM_WORDINFO - Informationen über die gefundenen Wörter. Zum Beispiel könnte eine Suche nach "a good book" die Werte "a: stopword, good:5637, book: 120" ausgeben.

- UDM_PARAM_SEARCHTIME - Suchzeit in Sekunden.
- UDM_PARAM_FIRST_DOC - Die Nummer des ersten auf der aktuellen Seite angezeigten Dokumentes.
- UDM_PARAM_LAST_DOC - Die Nummer des letzten auf der aktuellen Seiten angezeigten Dokumentes.

udm_load_ispell_data (PHP 4 >= 4.0.5)

Lade ISpell-Daten

int **udm_load_ispell_data** (int agent, int var, string val1, string val2, int flag)

udm_load_ispell_data() lädt ISpell-Daten. Gibt `TRUE` bei Erfolg, `FALSE` bei einem Fehler zurück.

`agent` - Zeiger auf einen Agenten, erhalten nach einem Aufruf von udm_alloc_agent().

`var` - Parameter, der die Quelle der ISpell-Daten angibt. Kann folgende Werte annehmen.

Nachdem diese Funktion aufgerufen wurde, sollte nicht vergessen werden, mittels udm_free_ispell_data() den durch ISpell belegten Speicher wieder freizugeben, auch wenn UDM_ISPELL_TYPE_SERVER verwendet wird.

Der schnellste Modus ist UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT ist langsamer, und UDM_ISPELL_TYPE_DB ist der langsamste. Dies gilt zumindest für mnoGoSearch 3.1.10 - 3.1.11, es ist jedoch geplant, den DB-Modus in zukünftigen Versionen schneller als den TEXT-Modus zu machen.

- UDM_ISPELL_TYPE_DB - Gibt an, das die ISpell-Daten aus einer SQL-Datenbank gelesen werden sollen. In diesem Fall werden *`val1`* und *`val2`* ignoriert und sollten leer gelassen werden. Der Parameter *`flag`* sollte `1` sein.

  Anmerkung: Der Parameter *`flag`* gibt an, dass die ISpell-Daten nach dem Laden sortiert werden sollen, dies ist für ein korrektes Arbeiten von ISpell nötig. Werden die ISpell-Daten aus Dateien gelesen, kann es vorkommen, daß mehrere Aufrufe von **udm_load_ispell_data()** vorkommen, was es wenig Effektiv macht, die Daten nach jedem Aufruf zu sortieren. Der Datenbank-Modus lädt jedoch alle Daten auf einmal, in diesem Fall sollte der Wert also `1` betragen. Sollte ein Fehler auftreten, wird `FALSE` zurückgegeben, Fehler und Fehlercode sind in diesem Fall über udm_error() und udm_errno() verfügbar.

  Beispiel:

```
if (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_DB,”,”,1)) {
printf("Error #%d: ’%s’\n",Udm_Errno($udm),Udm_Error($udm));
exit;
}
```

- UDM_ISPELL_TYPE_AFFIX - Gibt an, dass die ISpell-Daten aus einer Datei geladen werden sollen und initiiert das Laden der Anhangsdaten. In diesem Fall gibt der Parameter *`val1`* den zweibuchstabigen Sprachcode an, für den die Anhänge geladen werden sollen, *`val2`* gibt den

Dateipfad an. Wird kein absoluter Pfad angeben, werden die Dateien nicht in UDM_CONF_DIR, sondern in im zum aktuellen Pfad relativen gesucht. Kann die Datei nicht geladen werden, gibt die Funktion `FALSE` zurück, und eine Fehlermeldung wird ausgegeben. Die Fehlercodes können nicht über udm_error() oder udm_errno() ausgelesen werden. Zur Erlärung des Parameters *`flag`* schauen Sie bitte bei UDM_ISPELL_TYPE_DB nach.

Beispiel:

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1
exit;
}
```

Anmerkung: *`flag`* hat nur beim letzten Aufruf den Wert `1`.

- UDM_ISPELL_TYPE_SPELL - Gibt an, dass die ISpell-Daten aus einer Datei gelesen werden sollen, und initiiert das Laden des ISpell-Wörterbuches. In diesem Fall gibt der Parameter *`val1`* den zweibuchstabigen Sprachcode an, für den die Anhänge geladen werden sollen, *`val2`* gibt den Dateipfad an. Wird kein absoluter Pfad angeben, werden die Dateien nicht in UDM_CONF_DIR, sondern in im zum aktuellen Pfad relativen gesucht. Kann die Datei nicht geladen werden, gibt die Funktion `FALSE` zurück, und eine Fehlermeldung wird ausgegeben. Die Fehlercodes können nicht über udm_error() oder udm_errno() ausgelesen werden. Zur Erklärung des Parameters *`flag`* schauen Sie bitte bei UDM_ISPELL_TYPE_DB nach.

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0
   (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1
 exit;
 }
```

Anmerkung: *`flag`* hat nur beim letzten Aufruf den Wert`1`.

- UDM_ISPELL_TYPE_SERVER - schaltet den Spell-Server support ein. *`val1`* gibt den Host an, auf dem der Spell-Server läuft.. *`val2`* ist noch ohne Funktion, wird in Zukunft aber den Port des Spell-Servers angeben. *`flag`* wird nicht benötigt, da die Daten bereits sortiert im Spell-Server vorliegen.

  Der Spell-Server (Spelld) liest die Rechtschreibdaten aus einer seperaten Konfigurationsdatei (Standardmäßig /usr/local/mnogosearch/etc/spelld.conf), sortiert sie und speichert sie im Speicher.

  Die Funktion **udm_load_ispell_data()** in UDM_ISPELL_TYPE_SERVER lädt eigentlich nicht die ISpell-Daten, sondern legt nur die Serveradresse fest. Eigentlich wird dieser Server automatisch von udm_find() benutzt, wenn eine Suche durchgeführt wird. Sollte ein Fehler

auftreten, wird keine Fehlermeldung ausgegeben, eine ISpell-Konvertierung kann dann jedoch natürlich auch nicht stattfinden.

Anmerkung: Diese Funktion ist seit mnoGoSearch 3.1.12 verfügbar.

Beispiel:

```
if (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SERVER,”,”,1)) {
    printf("Error loading ispell data from server<br>\n");
    exit;
   }
```

udm_set_agent_param (PHP 4 >= 4.0.5)

Setzt die Parameter der aktuellen mnoGoSearch Session

int **udm_set_agent_param** (int agent, int var, string val)

udm_set_agent_param() gibt `TRUE` bei Erfolg, `FALSE` bei eine, Fehler zurück. Deklariert mnoGoSearch Sessionparameter.

Die folgenden Parameter und deren Werte sind erlaubt:

- UDM_PARAM_PAGE_NUM - wird genutzt, um eine bestimmte Seite aus den Suchergebnissen auszuwählen. Ergebnissseiten werden von 0 an gezählt.
- UDM_PARAM_PAGE_SIZE - Anzahl der Suchergebnisse, die pro Seite angezeigt werden.
- UDM_PARAM_SEARCH_MODE - Suchmodus. Die folgenden Werte sind erlaubt: UDM_MODE_ALL - Logische UND-Verknüpfung; UDM_MODE_ANY - Logische ODER-Verknüpfung; UDM_MODE_BOOL - boolsche Suche. Siehe udm_find() für weitere Details zur boolschen Suche.
- UDM_PARAM_CACHE_MODE - Schaltet den Ergebnis-Cache an oder aus. Falls eingeschaltet wird die Suchmaschine Suchergebnisse auf der Festplatte zwischenspeichern. Für den Fall, dass später eine ähnliche Suche durchgeführt wird, werden die Ergebnisse für ein schnelleres Ergebnis aus dem Cache gelesen. Werte: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - schaltet den Trackquery-Modus an oder aus. Seit Version 3.1.12 von mnoGoSearch wird eine Verfolgung von Suchanfragen unterstützt. Um die Track-Funktion nutzen zu können, müssen zusätzliche SQL-Tabellen angelegt werden. Für MySQL liegen die Tabellenbeschreibungen in create/mysql/track.txt. Wird eine Suche durchgeführt, werden die Suchwörter in diesen Tabellen zusammen mit der Anzahl der gefundenen Dokumente und dem aktuellen Unix-Timestamp gespeichert. Verfügbare Werte: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - Gibt an, ob die Index-Datenbank eine Unterstützung für "Phrases" erlaubt. (Muß vor der Indizierung in der indexer.conf eingeschaltet werden. Mögliche Werte: UDM_PHRASE_ENABLED und UDM_PHRASE_DISABLED. Wird der Phrase-Modus

verwendet, ist es natürlich nach wie vor möglich, alle anderen Suchmodi zu verwenden. Seit Version 3.1.11 wird auch der Cache-Modus zusammen mit dem Phrase-Modus unterstützt.

Beispiele:

"Arizona desert" - Diese Abfrage gibt alle Dokumente zurück, die "Arizona desert" als eine Zeichenkette enthalten. Für eine Suche nach kompletten Zeichenkette müssen diese in doppelte Anführungszeichen gesetzt werden.

- UDM_PARAM_CHARSET - Legt einen lokalen Zeichensatz wie koi8-r, cp1252 fest.
- UDM_PARAM_STOPFILE - Legt den Namen und den Pfad einer stopword-Datei fest. Auch hier wird nicht relativ zu UDM_CONF_DIR gesucht sondern relativ zu zu dem Pfad wo das aktuelle PHP-Script läuft.
- UDM_PARAM_STOPTABLE - Lädt die stopwords aus der angegebenen SQL-Tabelle. Es können mehrere Stopwordtable-Kommandos verwendet werden.
- UDM_PARAM_WEIGHT_FACTOR - Legt fest, wie bestimmte Teile eines Dokumentes zur Sortierung gewertet werden. Zur Zeit werden Body, Title, Keywords, Description und URL unterstützt. Um dieses Feature nutzen zu können, muß der Grad 2 in den *Weight-Anweisungen in der indexer.conf verwendet werden. Stellen wir uns vor, folgende Werte würden verwendet:

  URLWeight 1
  BodyWeight 2
  TitleWeight 4
  KeywordWeight 8
  DescWeight 16

  Solange der Indexer eine logische bit-ODER Operation verwendet, um das Gewicht einzelner Wörter festzulegen, wenn ein Wort mehrere Male im selben Dokument vorkommt, ist es möglich, zur Zeit der Suche festzustellen, welches Wort in welchem Teil des Dokumentes vorkommt. Wörter, die nur im Body auftauchen, haben das Gewicht 00000010 in binärer Notation. Wörter, die in allen Dokumententeilen vorkommen haben das Gewicht 00011111.

  Der Wert dieses Parameter ist ein String, bestehend aus den hexadezimalen Ziffern ABCDE. Jede Ziffer ist ein Faktor für das korrespondierende Bit. Für die oben angeführte Konfiguration zum Beispiel:

  E ist ein Faktor für das Gewicht 1 (URL Weight bit)
  D ist ein Faktor für das Gewicht 2 (BodyWeight bit)
  C ist ein Faktor für das Gewicht 4 (TitleWeight bit)
  B ist ein Faktor für das Gewicht 8 (KeywordWeight bit)
  A ist ein Faktor für das Gewicht 16 (DescWeight bit)

  Beispiele:

  UDM_PARAM_WEIGHT_FACTOR=00001 wird nur in URLs suchen.

  UDM_PARAM_WEIGHT_FACTOR=00100 wird nur in Titeln suchen.

  UDM_PARAM_WEIGHT_FACTOR=11100 wird in Titel, Keywords, Description aber nicht in URL und Body suchen.

  UDM_PARAM_WEIGHT_FACTOR=F9421 wird folgendermaßen Suchen und Werten:

  Beschreibung mit dem Faktor 15 (F hex)
  Keywords mit dem Faktor 9

Title mit dem Faktor 4
Body mit dem Faktor 2
URL mit dem Faktor 1

Wenn die Variable UDM_PARAM_WEIGHT_FACTOR übergeben wird, wird das Originalgewicht zur Sortierung verwendet.

- UDM_PARAM_WORD_MATCH - Wortteilübereinstimmung. Dieser Parameter kann verwendet werden, um festzulegen, welcher Wortteil mit dem gesuchten Wort übereinstimmen muß. Dieses Feature funktioniert nur im "single" oder "multi"-Modus. Im Cachemode und in den CRC-Modi ist eine Verwendung nicht möglich. Erlaubt Werte:

  UDM_MATCH_BEGIN - Der Wortanfang muss übereinstimmen;

  UDM_MATCH_END - Das Wortende muss übereinstimmen;

  UDM_MATCH_WORD - Das gesamte Wort muss übereinstimmen;

  UDM_MATCH_SUBSTR - Teile des Wortes müssen übereinstimmen.

- UDM_PARAM_MIN_WORD_LEN - Legt die mininmale Wortlänge fest. Jedes Wort, dass kürzer als die angegebene Länge ist, wird als stopword aufgefasst.
- UDM_PARAM_ISPELL_PREFIXES - Mögliche Werte: UDM_PREFIXES_ENABLED und UDM_PREFIXES_DISABLED, die explizit die Verwendung von ISpell-Prefixen an- oder abschalten. Wird also zum Beispiel das Wort "tested" in einer Suchabfrage verwendet, werden auch Wörter wie "test" oder "testing" gefunden. Nur Endungen werden normalerweise Unterstützt. Prefixe verändern für gewöhnlich die Bedeutung von Wörtern. So werden bei einer Suche nach "tested" z.B. auch Dokumente gefunden, die das Wort "untested" enthalten. Um diese Funktion nutzen zu können, müssen vorher mittels udm_load_ispell_data() ISpell-Daten geladen werden.
- UDM_PARAM_CROSS_WORDS - Aktiviert oder deaktiviert eine Unterstützung für Crosswords. Mögliche Werte: UDM_CROSS_WORDS_ENABLED und UDM_CROSS_WORDS_DISABLED.

  Das Crosswords-Feature erlaubt es, Wörter zwischen <a href="xxx"> und </a> dem Dokument zuzuordnen, zu dem dieser Link führt.

  Anmerkung: Crosswords werden seit mnoGoSearch 3.1.11 unterstützt.

- UDM_PARAM_VARDIR - Gibt einen Pfad zu einem Verzeichnis an, in dem der Indexer Daten speichert, wenn der integrierte Datenbank-Modus oder der Cachemodus verwendet wird. Voreingestellt ist das `/var`-Verzeichnis der mnoGoSearch-Installation verwendet. Es können ausschliesslich Strings verwendet werden. Dieser Parameter ist erst ab PHP 4.1.0 verfügbar.

udm_check_charset (PHP 4 CVS only)

Überprüft, ob mnoGoSearch der übergebene Zeichensatz bekannt ist

int **udm_check_charset** (int agent, string charset)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

udm_check_stored (PHP 4 CVS only)

Open connection to stored

int **udm_check_stored** (int agent, int link, string doc_id)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

udm_close_stored (PHP 4 CVS only)

Open connection to stored

int **udm_close_stored** (int agent, int link)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

udm_crc32 (PHP 4 CVS only)

Gibt eine CRC32-Prüfsumme eines übergebenen Strings zurück.

int **udm_crc32** (int agent, string str)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

udm_open_stored (PHP 4 CVS only)

Open connection to stored

int **udm_open_stored** (int agent, string storedaddr)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LX. mSQL Funktionen

Diese Funktionen erlauben Ihnen, auf einen mSQL-Datenbankserver zuzugreifen. Um über diese Funktionen verfügen zu können, müssen Sie PHP mit der mSQL-Unterstützung kompilieren, indem Sie die Option `--with-msql[=dir]` angeben. Das Standardverzeichnis ist /usr/local/Hughes.

Mehr Informationen über mSQL findet man bei http://www.hughes.com.au/ (http://www.hughes.com.au/).

msql (PHP 3, PHP 4 >= 4.0.0)

Sendet eine mSQL- Abfrage an den Datenbankserver

int **msql** (string database, string query [, int link_identifier])

Gibt eine positive Zahl als mSQL Ergebniskennung für diese Abfrage zurück oder `FALSE` bei einem Fehler

msql() wählt eine Datenbank aus und führt auf ihr eine Abfrage aus. Falls die optionale Verbindungskennung nicht angegeben ist, versucht die Funktion, eine offene Verbindung zum Datenbankserver zu finden. Wenn keine Verbindung gefunden wird, wird versucht, eine zu erzeugen, so als würde msql_connect() ohne Argumente aufgerufen. (siehe msql_connect()).

msql_affected_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt die Anzahl der betroffenen Datensätze zurück

int **msql_affected_rows** (int query_identifier)

Gibt die Anzahl Datensätze zurück, die von der letzten Abfrage betroffen ("berührt") wurden. (z. B. die Anzahl der Zeilen, die von einem SELECT zurückgeliefert wurden, die Anzahl der Zeilen, die durch einen UPDATE geändert wurden oder die Anzahl der Zeilen, die von einem DELETE gelöscht wurden.)

Siehe auch: msql_query().

msql_close (PHP 3, PHP 4 >= 4.0.0)

Schliesst eine mSQL-Verbindung

int **msql_close** (int link_identifier)

Gibt bei Erfolg `TRUE` zurück, bei einem Fehler `FALSE`.

msql_close() schliesst die mit der angegebenen Verbindungskennung verknüpfte Verbindung zum mSQL Datenbankserver. Falls die Verbindungskennung nicht angegeben ist, wird die zuletzt geöffnete Verbindung geschlossen.

Beachten Sie, dass diese Funktion normalerweise nicht notwendig ist, weil geöffnete nicht-persistente Verbindungen automatisch geschlossen werden, wenn die Ausführung des Skripts beendet ist.

msql_close() schliesst keine persistenten Verbindungen, die mit msql_pconnect() geöffnet wurden.

Siehe auch: msql_connect() und msql_pconnect().

msql_connect (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine mSQL Verbindung

int **msql_connect** ([string server [, string username [, string password]]])

Gibt bei Erfolg eine positive Zahl als Verbindungskennung zurück oder `FALSE` bei einem Fehler.

msql_connect() erzeugt eine Verbindung zu einem mSQL Datenbankserver. Die Angabe eines Hostnamens als Argument ist optional. Wenn Hostname fehlt, wird localhost vorausgesetzt.

Falls **msql_connect()** zum zweiten Mal mit denselben Argumenten aufgerufen wird, wird keine neue Verbindung erzeugt, stattdessen wird die Verbindungskennung der aktuell geöffneten Verbindung zurückgegeben.

Die Verbindung zum Server wird geschlossen, sobald die Ausführung des Skripts beendet ist, es sei denn, die Verbindung wurde zuvor durch einen expliziten Aufruf von msql_close() beendet.

Siehe auch msql_pconnect(), msql_close().

msql_create_db (PHP 3, PHP 4 >= 4.0.0)

Erzeugt eine mSQL Datenbank

int **msql_create_db** (string database name [, int link_identifier])

msql_create_db() erzeugt eine neue Datenbank auf dem mSQL-Server, die mit der angegebenen Verbindungskennung verknüpft ist.

Siehe auch: msql_drop_db().

msql_createdb (PHP 3, PHP 4 >= 4.0.0)

Erzeugt eine mSQL Datenbank

int **msql_createdb** (string database name [, int link_identifier])

Identisch mit msql_create_db().

msql_data_seek (PHP 3, PHP 4 >= 4.0.0)

Bewegt den internen Datensatzzeiger

int **msql_data_seek** (int query_identifier, int row_number)

Gibt bei Erfolg `TRUE` zurück, bei Fehler `FALSE`.

msql_data_seek() bewegt den internen Datensatzzeiger der mSQL Ergebnismenge, die mit der angegebenen Abfrage-ID verknüpft ist, zu der übergebenen Zeile. Der nächste Aufruf von msql_fetch_row() gibt diese Zeile zurück.

Siehe auch: msql_fetch_row().

msql_dbname (PHP 3, PHP 4 >= 4.0.0)

Gibt den aktuellen mSQL Datenbanknamen zurück

string **msql_dbname** (int query_identifier, int i)

msql_dbname() gibt den Namen der Datenbank zurück, die auf Position `i` in der Ergebnismenge der Funktion msql_listdbs() gespeichert ist. Die Funktion msql_numrows() kann benutzt werden, um zu sehen, wieviele Datenbanknamen verfügbar sind.

msql_drop_db (PHP 3, PHP 4 >= 4.0.0)

Löscht eine mSQL Datenbank

int **msql_drop_db** (string database_name, int link_identifier)

Gibt bei Erfolg `TRUE` zurück, bei Fehler `FALSE`.

msql_drop_db() löscht eine Datenbank, die mit der übergebenen Verbindungskennung verknüpft ist, vollständig vom Server.

Siehe auch: msql_create_db().

msql_dropdb (PHP 3, PHP 4 >= 4.0.0)

Löscht eine mSQL Datenbank

Siehe msql_drop_db().

msql_error (PHP 3, PHP 4 >= 4.0.0)

Gibt die letzte mSQL-Fehlermeldung zurück

string **msql_error** ( )

Fehler, die von mSQL zurückgeliefert werden, geben keine Warnungen mehr aus. Benutzen Sie diese Funktion, um den Error-String zu erhalten.

msql_fetch_array (PHP 3, PHP 4 >= 4.0.0)

Holt einen Datensatz als assoziatives Array

int **msql_fetch_array** (int query_identifier [, int result_type])

Gibt die Inhalte des gelesenen Datensatzes in einem assoziativen Array zurück oder `FALSE`, falls kein Datensatz mehr gelesen werden kann.

msql_fetch_array() ist eine erweiterte Version von msql_fetch_row(). Zusätzlich zur Speicherung der Daten in einem numerischen Array speichert diese Funktion den Datensatz mit assoziativen Indizes, indem sie die Feldnamen als Schlüssel benutzt.

Das zweite, optionale Argument `result_type` in **msql_fetch_array()** ist eine Konstante, die die folgenden Werte annehmen kann: MSQL_ASSOC, MSQL_NUM, und MYSQL_BOTH.

Seien Sie vorsichtig, wenn Sie ein Abfrageergebnis haben, das einen Datensatz zurückliefert, das nur ein Feld enthält, welches den Wert 0 (oder einen leeren String oder `NULL`) enthält.

Wichtig zu bemerken ist, **msql_fetch_array()** ist NICHT wesentlich langsamer als msql_fetch_row(), obwohl es einen zusätzlichen Ergebniswert zurückgibt.

Für weitere Einzelheiten siehe msql_fetch_row().

msql_fetch_field (PHP 3, PHP 4 >= 4.0.0)

Gibt Feldinformationen in einem Objekt zurück

object **msql_fetch_field** (int query_identifier, int field_offset)

Gibt Feldinformationen in einem Objekt zurück

msql_fetch_field() kann benutzt werden, um Informationen über Felder eines bestimmten Abfrageergebnisses in einem Objekt zu erhalten. Falls der Feldname nicht angegeben wurde, wird das nächste, noch nicht mit **msql_fetch_field()** geholte Feld zurückgegeben.

Die Eigenschaften dieses Objekts sind:

- name - Spaltenname
- table - Tabellenname, zu der die Spalte gehört
- not_null - 1 falls die Spalte keine NULL-Werte erlaubt
- primary_key - 1 falls das Feld ein Primärschlüssel ist
- unique - 1 falls die Spalte ein eindeutiger Schlüssel ist
- type - der Typ der Spalteninhalte

Siehe auch msql_field_seek().

msql_fetch_object (PHP 3, PHP 4 >= 4.0.0)

Holt einen Datensatz als Objekt

int **msql_fetch_object** (int query_identifier [, int result_type])

Gibt ein Objekt mit den Eigenschaften jeder Spalte der geholten Zeile eines Abfrageergebnisses zurück, oder `FALSE`, falls keine Zeilen mehr geholt werden können.

msql_fetch_object() ist ähnlich wie msql_fetch_array(), mit einem Unterschied - dass ein Objekt anstelle eines Arrays zurückgegeben wird. Indirekt bdeutet das, dass man auf die Daten nur über die

Feldnamen zugreifen kann, und nicht über ihre Feldnummern. (Ziffern sind unzulässige Namen für Eigenschaften.)

Das optionale zweite Argument *result_type* in msql_fetch_array() ist eine Konstante, die die folgenden Werte annehmen kann: MSQL_ASSOC, MSQL_NUM, und MSQL_BOTH.

In der Geschwindigkeit ist die Funktion identisch zu msql_fetch_array(), und fast so schnell wie msql_fetch_row() (der Unterschied ist unerheblich).

Siehe auch: msql_fetch_array() und msql_fetch_row().

msql_fetch_row (PHP 3, PHP 4 >= 4.0.0)

Holt einen Datensatz als numerisches Array

array **msql_fetch_row** (int query_identifier)

Gibt ein Array mit einem Element für jede Spalte des Datensatzes zurück, oder `FALSE`, falls kein Datensatz mehr geholt werden kann.

msql_fetch_row() holt einen Datensatz des Abfrageergebnisses, das mit dem Ergebnisbezeichner verknüpft ist. Der Datensatz wird als Array zurückgegeben. Alle Spalteninhalte des Ergebnisses werden unter einem numerischen Index, der bei 0 beginnt, abgelegt.

Aufeinanderfolgende Aufrufe von **msql_fetch_row()** geben jeweils die nächste Zeile des Abfrageergebnisses zurück, oder `FALSE`, wenn keine Datensätze mehr gelesen werden können.

Siehe auch: msql_fetch_array(), msql_fetch_object(), msql_data_seek(), und msql_result().

msql_fieldname (PHP 3, PHP 4 >= 4.0.0)

Gibt den Feldnamen zurück

string **msql_fieldname** (int query_identifier, int field)

msql_fieldname() Gibt den Namen des bezeichneten Feldes zurück. *query_identifier* ist der Ergebnisbezeichner, und *field* ist der Index des Feldes. `msql_fieldname($result, 2);` gibt den Namen des dritten Feldes des Abfrageergebnisses zurück, das mit dem Ergebnisbezeichner verknüpft ist.

msql_field_seek (PHP 3, PHP 4 >= 4.0.0)

Setzt den internen Feld-Zeiger auf den angegebenen Feldindex

int **msql_field_seek** (int query_identifier, int field_offset)

Setzt den internen Feld-Zeiger auf das angegebenen Feld. Falls der nächste Aufruf von msql_fetch_field() keinen Parameter für den Index enthält, wird dieses Feld zurückgegeben.

Siehe auch: msql_fetch_field().

msql_fieldtable (PHP 3, PHP 4 >= 4.0.0)

Gibt den Tabellennamen für ein Feld zurück

int **msql_fieldtable** (int query_identifier, int field)

Gibt den Namen der Tabelle zurück, von wo `field` das Feld gelesen wurde.

msql_fieldtype (PHP 3, PHP 4 >= 4.0.0)

Gibt den Datentyp des angegebenen Feldes zurück

string **msql_fieldtype** (int query_identifier, int i)

msql_fieldtype() ist ähnlich wie die Funktion msql_fieldname(). Die Argumente sind identisch, nur wird hier der Typ des Feldes zurückgegeben. Das kann "int", "char" oder "real" sein.

msql_fieldflags (PHP 3, PHP 4 >= 4.0.0)

Gibt alle Flags für das angegebene Feld zurück

string **msql_fieldflags** (int query_identifier, int i)

msql_fieldflags() gibt die Flags für das angegebene Feld zurück. Im Moment ist das entweder "not `NULL`", "primary key", eine Kombination der beiden Flags oder "" (ein leerer String).

msql_fieldlen (PHP 3, PHP 4 >= 4.0.0)

Gibt die Länge des angegebenen Feldes zurück

int **msql_fieldlen** (int query_identifier, int i)

msql_fieldlen() gibt die Länge des angegebenen Feldes zurück.

msql_free_result (PHP 3, PHP 4 >= 4.0.0)

Gibt den durch Ergebnisse belegten Speicher frei

int **msql_free_result** (int query_identifier)

msql_free_result() gibt den Speicher, den das mit `query_identifier` verknüpfte Ergebnis belegt hat, frei. Wenn PHP ein Skript beendet, wird dieser Speicher automatisch freigegeben, so dass man diese Funktion nur aufrufen muss, um sicher zu sein, nicht zuviel Speicherplatz während der Laufzeit des Skripts zu belegen.

msql_freeresult (PHP 3, PHP 4 >= 4.0.0)

Gibt den durch Ergebnisse belegten Speicher frei

Siehe msql_free_result()

msql_list_fields (PHP 3, PHP 4 >= 4.0.0)

Listet alle Felder einer Tabelle auf

int **msql_list_fields** (string database, string tablename)

msql_list_fields() liefert Informationen über die angegebene Tabelle. Argumente sind der Name der Datenbank und der Tabellenname. Ein Ergebnisbezeichner wird zurückgegeben, der mit msql_fieldflags(), msql_fieldlen(), msql_fieldname(), und msql_fieldtype() benutzt werden kann. Ein Ergebnisbezeichner ist eine positive Ganzzahl. Die Funktion gibt `-1` zurück, falls ein Fehler auftritt. Ein String, der den Fehler beschreibt, wird in `$phperrmsg` geschrieben und auf dem Bildschirm ausgegeben, es sei denn, die Funktion wurde als `@msql_list_fields()` aufgerufen.

Siehe auch msql_error().

msql_listfields (PHP 3, PHP 4 >= 4.0.0)

Listet alle Felder einer Tabelle auf

Siehe msql_list_fields().

msql_list_dbs (PHP 3, PHP 4 >= 4.0.0)

Listet alle mSQL Datenbanken auf dem Server auf

int **msql_list_dbs** (void)

msql_list_dbs() gibt einen Ergebnisbezeichner zurück, der Namen aller Datenbanken, die auf dem mSQL-Server liegen, enthält. Mit der Funktion msql_dbname() kann dieser Ergebnisbezeichner durchlaufen werden.

msql_listdbs (PHP 3, PHP 4 >= 4.0.0)

Listet alle mSQL Datenbanken auf dem Server auf

Siehe msql_list_dbs().

msql_list_tables (PHP 3, PHP 4 >= 4.0.0)

Listet alle Tabellen einer mSQL-Datenbank auf

int **msql_list_tables** (string database)

msql_list_tables() hat als Argumente einen Datenbanknamen und liefert Ergebnisbezeichner, fast gleich, wie die Funktion msql(). Die Funktion msql_tablename() sollte benutzt werden, um die aktuellen Tabellennamen aus dem Ergebnisbezeichner zu erhalten.

msql_listtables (PHP 3, PHP 4 >= 4.0.0)

Listet alle Tabellen einer mSQL-Datenbank auf

Siehe msql_list_tables().

msql_num_fields (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl der Felder eines Abfrageergebnisses zurück

int **msql_num_fields** (int query_identifier)

msql_num_fields() Gibt die Anzahl der Felder eines Abfrageergebnisses zurück.

Siehe auch: msql(), msql_query(), msql_fetch_field(), und msql_num_rows().

msql_num_rows (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl der Zeilen eines Abfrageergebnisses zurück

int **msql_num_rows** (int query_identifier)

msql_num_rows() Gibt die Anzahl der Zeilen eines Abfrageergebnisses zurück.

Siehe auch: msql(), msql_query(), und msql_fetch_row().

msql_numfields (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl der Felder eines Abfrageergebnisses zurück

int **msql_numfields** (int query_identifier)

Identisch zu msql_num_fields().

msql_numrows (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl der Zeilen eines Abfrageergebnisses zurück

int **msql_numrows** (void)

Identisch zu msql_num_rows().

msql_pconnect (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine persistente mSQL Verbindung

int **msql_pconnect** ([string server [, string username [, string password]]])

Gibt bei Erfolg eine positive Zahl als mSQL Verbindungskennung für die persistente Verbindung zurück, oder FALSE bei einem Fehler.

msql_pconnect() verhält sich sehr ähnlich wie msql_connect(), mit zwei grossen Unterschieden.

Erstens, wenn die Verbindung aufgebaut wird, prüft die Funktion zuerst, ob eine geöffnete persistente Verbindung zu demselben Host besteht. Wird eine Verbindung gefunden, wird deren Verbindungskennung zurückgegeben anstatt eine neue Verbindung aufzubauen.

Zweitens, die Verbindung zum mSQL-Server wird nicht geschlossen, wenn die Ausführung des Skripts beendet ist. Stattdessen bleibt die Verbindung für den späteren Gebrauch offen. (msql_close() schliesst keine Verbindungen, die mit **msql_pconnect()** erzeugt wurden).

Darum wird diese Verbindungsart als 'persistent' bezeichnet.

msql_query (PHP 3, PHP 4 >= 4.0.0)

Schickt eine mSQL-Abfrage zum Server

int **msql_query** (string query, int link_identifier)

msql_query() schickt eine Datenbankabfrage an die aktuelle Datenbank auf dem Server, die mit der angegebenen Verbindungskennung verknüpft ist. Wenn keine Verbindungskennung angegeben ist, wird die zuletzt geöffnete Verbindung angenommen. Falls keine Verbindung geöffnet ist, versucht die Funktion, eine Verbindung aufzubauen, die sie so benutzt, als wäre msql_connect() aufgerufen worden.

Gibt bei Erfolg eine positive Zahl als mSQL Verbindungskennung zurück, FALSE bei einem Fehler.

Siehe auch: msql(), msql_select_db(), und msql_connect().

msql_regcase (PHP 3, PHP 4 >= 4.0.0)

Übersetzt einen case-sensitiven regulären Ausdruck in einen insensitiven Ausdruck

Siehe sql_regcase().

msql_result (PHP 3, PHP 4 >= 4.0.0)

Holt den Wert eines einzelnen Feldes der übergebenen Zeile

int **msql_result** (int query_identifier, int i, mixed field)

Gibt den Wert zurück, der in der übergebenen Zeile und Spalte des bezeichneten mSQL-Abfrageergebnisses steht.

msql_result() gibt den Inhalt einer Tabellenzelle des mSQL-Abfrageergebnisses zurück. Das field Argument kann sowohl der Index als auch der Feldname des Feldes sein, oder, ein mit dem Tabellennamen qualifizierter Feldname (Tabellenname.Feldname). Falls der Spaltenname ein Alias ist ('select foo as bar from ...'), wird der Aliasname anstatt des Feldnamens angegeben.

Wenn man mit grossen Abfrageergebnissen arbeitet, sollte man die Benutzung der Funktionen in Betracht ziehen, die ganze Zeilen aus der Tabelle auslesen (wie unten beschrieben). Weil diese Funktionen die Inhalte von verschiedene Tabellenzellen in einem Funktionsaufruf holen, sind sie viel schneller als **msql_result()**. Beachten sie auch, dass die Angabe eines numerischen Index für das field Argument viel schneller ist, als die Angabe eines Feldnamens oder eines Tabellenname.Feldname Arguments.

Empfohlene Alternativen mit hoher Ausführungsgeschwindigkeit: msql_fetch_row(), msql_fetch_array(), und msql_fetch_object().

msql_select_db (PHP 3, PHP 4 >= 4.0.0)

Wählt eine mSQL Datenbank aus

int **msql_select_db** (string database_name, int link_identifier)

Gibt bei Erfolg `TRUE` zurück, `FALSE` bei einem Fehler.

msql_select_db() wählt die aktuell aktive Datenbank, die mit der übergebenen Verbindungskennung verknüpft ist, auf dem Server aus. Falls keine Verbindungskennung angegeben wurde, wird die zuletzt geöffnete Verbindung vorausgesetzt. Falls keine Verbindung geöffnet ist, versucht die Funktion eine Verbindung aufzubauen und zu benutzen, so als wäre msql_connect() aufgerufen worden.

Jeder nachfolgende Aufruf von msql_query() bezieht sich auf die aktive Datenbank.

Siehe auch: msql_connect(), msql_pconnect(), und msql_query().

msql_selectdb (PHP 3, PHP 4 >= 4.0.0)

Wählt eine mSQL Datenbank aus

Siehe msql_select_db().

msql_tablename (PHP 3, PHP 4 >= 4.0.0)

Gibt den Tabellennamen für ein Feld zurück

string **msql_tablename** (int query_identifier, int field)

msql_tablename() hat als Argumente einen Ergebnisbezeichner, der von der Funktion msql_list_tables() zurückgegeben wurde, und einen Integerwert und gibt den Namen einer Tabelle zurück. Die Funktion msql_numrows() kann benutzt werden, um die Anzahl der Tabellen in einem Ergebnisbezeichner festzustellen.

Beispiel 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("wisconsin");
$i = 0;
while ($i < msql_numrows ($result)) {
    $tb_names[$i] = msql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

LXI. MySQL Funktionen

Diese Funktionen erlauben den Zugriff auf einen MySQL Datenbank-Server. Um diese Funktionen nutzen zu können, müssen Sie PHP mit MySQL Unterstützung übersetzen. Das erreichen Sie durch die Angabe der Option `--with-mysql` beim Übersetzen. Falls Sie den Pfad zu MySQL nicht angeben, verwendet PHP die internen MySQL Client Bibliotheken. Benutzer die MySQL auch in anderen Anwendungen nutzen (zum Beispiel: PHP3 und PHP4 als simultane Apache-Module oder auth-mysql) sollten immer den Pfad zu MySQL mit der Option `--with-mysql=/pfad/zu/mysql` festlegen. Diese Angabe zwingt PHP die Client-Bibliotheken zu nutzen, die von MySQL installiert wurden. Diese Vorgehensweise vermeidet jegliche Konflikte.

Weitere Informationen zu MySQL gibt es unter http://www.mysql.com/.

Dokumentation zu MySQL finden Sie unter http://www.mysql.com/documentation/.

In diesem Dokument wird eine Reihe von Begriffen benutzt, die hier kurz erläutert werden sollen.

Anfrageergebnis, Ergebnis

Die Rückgabe auf eine Anfrage an den Datenbankserver. Anfragen können sowohl Datenbankanfragen sein, die Teile des Datenbankinhalts umfassen oder Informationen über eine Datenbank oder den Datenbankserver liefern. Ist das Ergebnis eine Datenbankanfrage kann man es sich als Tabelle vorstellen, in der die Zeilen die Datensätze sind und die Felder den Spalten ensprechen. Ein Ergebnis besteht aus 0 oder beliebig vielen Datensätzen.

Ergebnis-Kennung

Eine Referenz auf ein Ergebnis. Über diese Kennung erfolgt grundsätzlich der Zugriff auf das Ergebnis.

Datensatz

Eine Zeile aus einem Anfrageergebnis bestehend aus den Werten der einzelnen Felder.

Datensatzzeiger

Ein interner Zeiger auf einen Datensatz in einem Anfrageergebnis. Dieser Zeiger bestimmt auf welchen Datensatz zugegriffen wird. Der Zeiger kann sowohl implizit als auch explizit verändert werden.

Feld

Entspricht einer Spalte im Anfrageergebnis.

Feldzeiger

Wie Datensatzzeiger bei Zugriffen auf Felder eines Anfrageergebnisses.

Verbindungs-Kennung

Eine Referenz auf eine Verbindung zum Datenbank-Server. Mit dieser Verbindung ist beispielsweise die Datenbank, der Benutzer, der Rechnername auf dem die Datenbank läuft und weiteres verknüpft. Jeder Zugriff auf den Server nutzt diese Kennung. Wenn die Kennung beim Aufruf einer Funktion nicht angegeben wird, so wird die aktuelle Verbindungs-Kennung verwendet, die intern von PHP verwaltet wird.

Das Verhalten der MySQL Funktionen wird durch Einträge in der globalen Konfigurationsdatei

beeinflusst.

Tabelle 1. MySQL Konfigurations-Optionen

Name	Standard	Veränderbar
mysql.allow_persistent	"On"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL

Für weitere Details und die Definition der PHP_INI_* Konstanten schauen Sie bei **ini_set()** nach.

Folgendes einfache Beispiel zeigt Ihnen, wie Sie sich mit einer MySQL- Datenbank verbinden, eine Anfrage ausführen, die Ergebnisse ausgeben und die Verbindung wieder trennen.

Beispiel 1. MySQL Erweiterung: ein Überblicks Beispiel

```
<?php
// Verbindung aufbauen, auswählen einer Datenbank
$link = mysql_connect("mysql_host", "mysql_login", "mysql_password")
    or die("Keine Verbindung möglich!");
print "Verbindung zum Datenbankserver erfolgreich";
mysql_select_db("Meine_Datenbank")
    or die("Auswahl der Datenbank fehlgeschlagen");

// ausführen einer SQL Anfrage
$query = "SELECT * FROM Meine_Tabelle";
$result = mysql_query($query)
    or die("Anfrage fehlgeschlagen");

// Ausgabe der Ergebnisse in HTML
print "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    print "\t<tr>\n";
    foreach ($line as $col_value) {
        print "\t\t<td>$col_value</td>\n";
    }
    print "\t</tr>\n";
}
print "</table>\n";

// Freigeben des Resultsets
mysql_free_result($result);

// schliessen der Verbinung
mysql_close($link);
?>
```

mysql_affected_rows (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl betroffener Datensätze einer vorhergehenden MySQL Operation

int **mysql_affected_rows** ([resource Verbindungs-Kennung])

mysql_affected_rows() liefert die Anzahl betroffener Datensätze durch die letzte INSERT, UPDATE oder DELETE Anfrage an den Server, die mit der angegebenen `Verbindungs-Kennung` assoziiert wird. Wird die Verbindungskennung nicht angegeben, wird die letzte durch mysql_connect() geöffnete Verbindung angenommen.

> **Anmerkung:** Falls Sie Transaktionen verwenden, müssen Sie **mysql_affected_rows()** unmittelbar nach Ihrer INSERT, UPDATE, oder DELETE Anfrage aufrufen, nicht nach dem Commit.

War die letzte Anfrage ein DELETE-Anweisung ohne WHERE Bedingung, wurden alle Datensätze aus der Tabelle gelöscht. Diese Funktion wird aber trotzdem `NULL` zurückliefern.

> **Anmerkung:** Benutzen Sie UPDATE wird MySQL keine Spalten aktualisieren, bei denen der neue dem alten Wert entpspricht. Das kann dazu führen, dass **mysql_affected_rows()** nicht die tatsächliche Anzahl der betroffenen Zeilen liefert, sondern nur die, die wörtlich durch die Anfrage betroffen sind.

Diese Funktion arbeitet nicht in Verbindung mit SELECT-Anfragen, sondern nur bei Anweisungen, die Datensätze verändern. Um die Anzahl der gelieferten Datensätze einer SELECT-Anfrage zu erhalten, benutzen Sie die Funktion mysql_num_rows().

Falls die letze Anfrage fehlschlägt, liefert diese Funktion -1 zurück.

Siehe auch: mysql_num_rows().

mysql_change_user (PHP 3>= 3.0.13)

Ändert den zur Zeit angemeldeten Benutzer der aktiven Verbindung

int **mysql_change_user** (string Benutzername, string Benutzerkennwort [, string Datenbank [, resource Verbindungs-Kennung]])

mysql_change_user() ändert den angemeldeten Benutzer der aktuellen, aktiven Datenbank-Verbindung, oder für die Verbindung, die mit dem optionalen Parameter `Verbindungs-Kennung` bestimmt wurde. Wurde eine `Datenbank` angegeben, wird diese zur aktiven Datenbank, nachdem der Benutzer gewechselt hat.

> **Anmerkung:** Diese Funktion wurde mit PHP 3.0.13 eingeführt und erfordert MySQL 3.23.3 oder höher.

mysql_close (PHP 3, PHP 4 >= 4.0.0)

Schließt eine Verbindung zu MySQL

bool **mysql_close** ([resource Verbindungs-Kennung])

Rückgabewert: Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

mysql_close() schließt die Verbindung zum MySQL-Server, die mit der angegebenen Verbindungs-Kennung verknüpft ist. Wird die *`Verbindungs-Kennung`* nicht angegeben, wird die zuletzt geöffnete Verbindung geschlossen.

Die Verwendung von **mysql_close()** ist für gewöhnlich nicht notwendig, weil offene, nicht persistente Verbindungen automatisch mit Beendigung des PHP-Skripts geschlossen werden. Siehe auch Freigabe von Resourcen.

Anmerkung: mysql_close() schließt keine persistenten Verbindungen, die mit mysql_pconnect() geöffnet wurden.

Beispiel 1. Verbindung zum Datenbankserver schließen

```
<?php
    $link = mysql_connect ("localhost", "marliesle", "secret")
        or die ("keine Verbindung möglich");
    print ("Verbindung erfolgreich");
    mysql_close ($link);
?>
```

Siehe auch: mysql_connect() und mysql_pconnect().

mysql_connect (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine Verbindung zu einem MySQL-Server

resource **mysql_connect** ([string Server [, string Benutzername [, string Benutzerkennwort]]])

Rückgabewert: Eine MySQL Verbindungs-Kennung im Erfolgsfall oder `FALSE` im Fehlerfall.

mysql_connect() öffnet eine Verbindung zu einem MySQL Server. Für fehlende optionale Parameter werden folgende Standardwerte angenommen: *`server`* = 'localhost:3306', *`Benutzername`* = Name des Benutzers dem der Server-Prozess gehört und *`Benutzerkennwort`* = leeres Kennwort.

Der Parameter *`Server`* kann zudem die Portnummer enthalten, z.B. "hostname:port" oder den Pfad zu einem Socket z.B. ":/pfad/zum/socket" für Zugriffe auf dem lokalen Rechner.

Anmerkung: Unterstützung für ":port" wurde in PHP 3.0.10 hinzugefügt.

Unterstützung für "::/pfad/zum/socket" wurde in PHP 3.0.10 hinzugefügt.

Eine Fehlermeldung beim Verbindungsaufbau kann unterdrückt werden, wenn dem Funktionsnamen ein '@' voran gestellt wird.

Für den Fall, dass ein zweiter Aufruf von **mysql_connect()** mit den gleichen Argumenten erfolgt, wird keine neue Verbindung aufgebaut, sondern die Verbindungs-Kennung der schon bestehenden Verbindung zurückgeliefert.

Die Verbindung zum Datenbank-Server wird geschlossen, sobald die Ausführung des PHP-Skripts beendet ist oder vorher explizit die Funktion mysql_close() aufgerufen wird.

Beispiel 1. Verbindung zum Datenbanksserver öffnen

```
<?php
    $link = mysql_connect ("localhost", "marliesle", "secret")
        or die ("keine Verbindung möglich");
    print ("Verbindung erfolgreich");
    mysql_close ($link);
?>
```

Siehe auch mysql_pconnect() und mysql_close().

mysql_create_db (PHP 3, PHP 4 >= 4.0.0)

Anlegen einer Datenbank

bool **mysql_create_db** (string Datenbankname [, resource Verbindungs-Kennung])

mysql_create_db() versucht eine neue Datenbank auf dem Server anzulegen, der mit der angegebenen Verbindungs-Kennung assoziiert wird.

Rückgabewert: Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Beispiel 1. Anlegen einer Datenbank

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim")
        or die ("keine Verbindung möglich");
    if (mysql_create_db ("meine_db")) {
        print ("Datenbank erfolgreich angelegt\n");
    } else {
        printf ("Fehler beim Anlegen der Datenbank: %s\n", mysql_error ());
    }
?>
```

Für Abwärtskompatibilität kann auch **mysql_createdb()** verwendet werden. Diese Funktion ist jedoch veraltet.

Siehe auch: mysql_drop_db().

mysql_data_seek (PHP 3, PHP 4 >= 4.0.0)

Bewegt den internen Ergebnis-Zeiger

bool **mysql_data_seek** (resource Ergebnis-Kennung, int Datensatznummer)

Rückgabewert: Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

mysql_data_seek() bewegt den internen Datensatz-Zeiger eines Anfrageergebnisses zum Datensatz mit der übergebenen Nummer. Der nächste Aufruf von mysql_fetch_row() liefert den entsprechenden Datensatz.

Die *`Datensatznummer`* beginnt bei 0.

Beispiel 1. Beispiel: Positionieren des Datensatzzeigers

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim")
        or die ("Keine Verbindung möglich");

    mysql_select_db ("samp_db")
        or die ("Auswahl der Datenbank nicht möglich");

    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_query ($query)
        or die ("Anfrage fehlgeschlagen");

    // holen der Zeilen in umgekehrter Reihenfolge

    for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!mysql_data_seek ($result, $i)) {
            echo "Cannot seek to row $i\n";
            continue;
        }

        if(!($row = mysql_fetch_object ($result)))
            continue;

        echo "$row->last_name $row->first_name<br />\n";
    }

    mysql_free_result ($result);
?>
```

mysql_db_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert Ergebnisdaten

string **mysql_db_name** (resource Ergebnis-Kennung, int Datensatz [, mixed Feld])

mysql_db_name() erwartet als ersten Parameter eine Ergebnis-Kennung eines vorhergehenden Aufrufs der Funktion mysql_list_dbs(). Der Parameter *Datensatz* ist ein Index innerhalb der Ergebnismenge.

Falls ein Fehler auftritt ist der Rückgabewert FALSE. Um die Art des Fehlers zu bestimmen, benutzen Sie die Funktionen mysql_errno() und mysql_error().

Beispiel 1. mysql_db_name() Beispiel

```
<?php
error_reporting(E_ALL);

mysql_connect('dbhost', 'username', 'password');
$db_list = mysql_list_dbs();

$i = 0;
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
    echo mysql_db_name($db_list, $i) . "\n";
    $i++;
}
?>
```

Für Abwärtskompatibilität kann auch **mysql_dbname()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_db_query (PHP 3, PHP 4 >= 4.0.0)

Absetzen einer Anfrage an die Datenbank

resource **mysql_db_query** (string Datenbank, string Anfrage [, resource Verbindungs-Kennung])

Rückgabewert: Eine positive MySQL Ergebnis-Resorce für das Anfrageergebnis, oder FALSE im Fehlerfall.

msql_db_query() bestimmt eine Datenbank und führt eine Anfrage an diese aus. Falls der optionale Parameter *Verbindungs-Kennung* nicht angegeben ist, wird versucht, eine offene Verbindung zum MySQL Server zu finden und zu nutzen. Wenn dies fehlschlägt, wird versucht eine Verbindung aufzubauen, auf die gleiche Art und Weise wie beim Aufruf von mysql_connect() ohne Angabe von Argumenten.

Siehe auch: mysql_connect() und mysql_query().

Anmerkung: Diese Funktion ist seit PHP 4.0.6 veraltet. Verwenden Sie diese Funktion nicht. Stattdessen sollten Sie die Funktionen mysql_select_db() und mysql_query() verwenden.

mysql_drop_db (PHP 3, PHP 4 >= 4.0.0)

Löschen einer Datenbank

bool **mysql_drop_db** (string Datenbankname [, resource Verbindungs-Kennung])

Rückgabewert: Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

mysql_drop_db() versucht eine komplette Datenbank vom Server, der mit der übergebenen *`Verbindungs-Kennung`* assoziiert wird, zu löschen.

Zur Abwärtskompatibilität kann auch die Funktion **mysql_dropdb()** verwendet werden. Diese Funktion ist jedoch veraltet.

Siehe auch: mysql_create_db().

mysql_errno (PHP 3, PHP 4 >= 4.0.0)

Liefert die Nummer einer Fehlermeldung einer zuvor ausgeführten MySQL Operation

int **mysql_errno** ([resource Verbindungs-Kennung])

Liefert die Fehlernummer der letzten MySQL Funktion, oder `0` (Null) falls kein Fehler aufgetreten ist.

Fehler, die vom MySQL Server kommen, führen nicht mehr zu einer Ausgabe von Warnungen. Stattdessen sollten Sie die Funktion **mysql_errno()** verwenden, um die Fehlernummer zu erhalten. Beachten Sie, dass diese Funktion nur die Fehlernummer der zuletzt ausgeführten MySQL Funktion liefert (ausgeschlossen sind die Funktionen mysql_error() und **mysql_errno()**). Wenn Sie diese Funktion verwenden wollen, sollten Sie den Wert überprüfen, bevor Sie eine weitere MySQL Funktion aufrufen.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("dbnichtvorhanden");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nichtvorhandenetabelle");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Siehe auch: mysql_error()

mysql_error (PHP 3, PHP 4 >= 4.0.0)

Liefert den Fehlertext der zuvor ausgeführten MySQL Operation

string **mysql_error** ([resource Verbindungs-Kennung])

Rückgabewert: Liefert den Fehlertext der letzten MySQL Funktion, oder ″ (eine leere Zeichenkette), falls kein Fehler aufgetreten ist.

Fehler, die vom MySQL Server kommen, führen nicht mehr zu einer Ausgabe von Warnungen. Stattdessen, sollten Sie die Funktion **mysql_error()** verwenden, um den Fehlertext zu erhalten. Beachten Sie, dass diese Funktion nur den Fehlertext der zuletzt ausgeführten MySQL Funktion liefert (ausgeschlossen sind die Funktionen **mysql_error()** und mysql_errno()). Wenn Sie diese Funktion verwenden wollen, sollten Sie den Wert überprüfen, bevor Sie eine weitere MySQL Funktion aufrufen.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("dbnichtvorhanden");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM tabellenichtvorhanden");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Siehe auch: mysql_errno()

mysql_escape_string (PHP 4 >= 4.0.3)

Maskiert einen String zur Benutzung in mysql_query.

string **mysql_escape_string** (string unescaped_string)

Diese Funktion maskiert *`unescaped_string`* zur sicheren Benutzung in mysql_query().

Anmerkung: Die Zeichen `%` und _ werden von **mysql_escape_string()** nicht maskiert.

mysql_fetch_array (PHP 3, PHP 4 >= 4.0.0)

Liefert einen Datensatz als assoziatives Array, als numerisches Array oder beides

array **mysql_fetch_array** (resource Ergebnis-Kennung [, int Ergebnistyp])

Liefert ein Array das dem aktuellen Datensatz entspricht oder `FALSE`, wenn keine weiteren Datensätze vorliegen.

mysql_fetch_array() ist eine erweiterte Version von mysql_fetch_row(). Die Daten werden sowohl unter numerischen Indizes des Ergebnis-Arrays abgelegt, als auch unter assoziativen Indizes. Als Schlüssel für die assoziativen Indizes werden die Feldnamen benutzt.

Falls zwei oder mehrere Spalten des Ergebnisses den gleichen Feldnamen haben, dann wird nur der Wert der letzten Spalte im Array unter diesem Feldnamen abgelegt. Um auch auf die anderen gleichnamigen Spalten zugreifen zu können, müssen Sie den numerischen Index der Spalte verwenden oder einen Alias für die Spalte vergeben. Falls Sie Aliase für Spalten verwenden, können Sie auf die Inhalte dieser Spalten nicht über ihren ursprünglichen Namen zugreifen. (In folgendem Beispiel können Sie `'field'` nicht benutzen).

```
select tone.field as foo ttwo.field as bar from tone, ttwo
```

Zu betonen ist, dass der Gebrauch von **mysql_fetch_array()** *nicht signifikant* langsamer ist als mysql_fetch_row(), obwohl die Funktion einen sichtlichen Mehrwert bietet.

Das optionale zweite Argumente *`Ergebnistyp`* in **mysql_fetch_array()** ist eine Konstante und kann die folgenden Werte annehmen: MYSQL_ASSOC, MYSQL_NUM und MYSQL_BOTH. Diese Eigenschaft wurde in PHP 3.0.7 hinzugefügt. MYSQL_BOTH ist der Standard für diesen Paramater.

Verwenden Sie MYSQL_BOTH, erhalten Sie ein Array mit assoziativen und numerischen Indizes. Bei MYSQL_ASSOC erhalten Sie nur aasoziative Indizes (genau wie bei mysql_fetch_assoc()). Bei MYSQL_NUM erhalten Sie nur numerische Indizes (genau wie bei mysql_fetch_row()).

Für weitere Details, siehe auch mysql_fetch_row() und mysql_fetch_assoc()

Beispiel 1. mysql_fetch_array() Beispiel

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database","select user_id, fullname from table");
while ($row = mysql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
mysql_free_result ($result);
?>
```

mysql_fetch_assoc (PHP 4 >= 4.0.3)

Liefert einen Datensatz als assoziatives Array

array **mysql_fetch_assoc** (resource Ergebnis)

Liefert ein assoziatives Array, das dem geholten entprechenden Datensatz enthält.Sind keine weiteren Datensaätze vorhanden gibt diese Funktion `FALSE`zurück.

mysql_fetch_assoc() entspricht in der Funktionsweise exakt dem Aufruf von mysql_fetch_array() mit Angabe von MYSQL_ASSOC als optionalen zweiten Parameter. Diese Funktion liefert Ihnen nur ein assoziatives Array. Das war das ursprüngliche Verhalten von mysql_fetch_array(). Falls Sie sowohl numerische, als auch assoziative Indizes benötigen, verwenden Sie mysql_fetch_array().

Falls zwei oder mehrere Felder des Ergebnisses den gleichen Feldnamen haben, dann wird nur der Wert des letzten Felds im Array unter diesem Feldnamen abgelegt. Um auch auf die anderen, gleichnamigen, Felder zugreifen zu können, müssen Sie entweder numerische Indizes und damit mysql_fetch_row() benutzen, oder Aliase für Ihre Felder anlegen. Zur Verwendung von Aliasen schauen Sie sich das Beispiel unter der Beschreibung von mysql_fetch_array() an.

Zu betonen ist, dass der Gebrauch von **mysql_fetch_assoc()** *nicht signifikant* langsamer ist als mysql_fetch_row(), obwohl die Funktion einen sichtlichen Mehrwert bietet.

Für weitere Details, siehe auch mysql_fetch_row() und mysql_fetch_array().

Beispiel 1. mysql_fetch_assoc() Beispiel

```
<?php
mysql_connect ($host, $user, $password);
mysql_select_db("database");
$result = mysql_query("select user_id, fullname from table");
while ($row = mysql_fetch_array ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result ($result);
?>
```

mysql_fetch_field (PHP 3, PHP 4 >= 4.0.0)

Liefert ein Objekt mit Feldinformationen aus einem Anfrageergebnis

object **mysql_fetch_field** (resource Ergebnis-Kennung [, int Feld-Offset])

Rückgabe: Ein Objekt mit Feldinformationen.

mysql_fetch_field() kann benutzt werden, um Informationen über Felder in einem bestimmten Anfrageergebnis zu erhalten. Wenn der Feld-Offset nicht angegeben wird, werden die Information des nächsten, noch nicht mit **mysql_fetch_field()** erfragten Feldes, geliefert.

Die Eigenschaften des Objekts sind:

- name - Feldname
- table - Name der Tabelle zu der das Feld gehört
- max_length - maximale Länge des Feldes
- not_null - 1, wenn das Feld nicht `NULL` sein kann
- primary_key - 1, wenn das Feld ein 'primary key' ist
- unique_key - 1, wenn das Feld ein 'unique key' ist

- multiple_key - 1, wenn das Feld ein ’non-unique key’ ist
- numeric - 1, wenn das Feld vom Typ ’numeric’ ist
- blob - 1, wenn das Feld vom Typ ’BLOB’ ist
- type - der Typ des Feldes
- unsigned - 1, wenn das Feld vorzeichenlos ist
- zerofill - 1, wenn das Feld ’zero-filled’ ist

Beispiel 1. mysql_fetch_field()

```
<?php
mysql_connect ($host, $user, $password)
    or die ("keine Verbindung möglich");
$result = mysql_db_query ("database", "select * from table")
    or die ("Anfrage fehlgeschlagen");
// get column metadata
$i = 0;
while ($i < mysql_num_fields ($result)) {
    echo "Information für Feld $i:<BR>\n";
    $meta = mysql_fetch_field ($result);
    if (!$meta) {
        echo "keine Information verfügbar<BR>\n";
    }
    echo "<PRE>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</PRE>";
    $i++;
}
mysql_free_result ($result);
?>
```

Siehe auch mysql_field_seek().

mysql_fetch_lengths (PHP 3, PHP 4 >= 4.0.0)

Liefert die Länge eines jeden Feldes in einem Ergebnis

array **mysql_fetch_lengths** (resource Ergebnis-Kennung)

Liefert ein Array, das die Länge eines jeden Feldes in dem zuletzt mit mysql_fetch_row() geholten Datensatz enthält, oder `FALSE` im Fehlerfall.

mysql_fetch_lengths() speichert die Länge jedes Ergebnisfelds in dem mit mysql_fetch_row(), mysql_fetch_array() und mysql_fetch_object() zuletzt geholten Datensatz in einem Array, dessen Offset bei 0 beginnt.

Siehe auch: mysql_fetch_row().

mysql_fetch_object (PHP 3, PHP 4 >= 4.0.0)

Liefert eine Ergebniszeile als Objekt

object **mysql_fetch_object** (resource Ergebnis-Kennung [, int Ergebnistyp])

Liefert ein Objekt mit Eigenschaften, die den Feldern des geholten Datensatzes entsprechen, oder `FALSE` wenn keine weiteren Datensätze vorhanden sind.

mysql_fetch_object() ähnelt mysql_fetch_array(), mit einem Unterschied - ein Objekt wird zurück geliefert anstatt eines Arrays. Indirekt bedeutet dies, dass Sie die Daten nur mit ihren Feldnamen und nicht mit dem Offset ansprechen können (Nummern sind ungültige Namen für Eigenschaften).

Das optionale zweite Argumente *`Ergebnistyp`* ist eine Konstante und kann einen der folgenden Werte annehmen: MYSQL_ASSOC, MYSQL_NUM und MYSQL_BOTH. Die Bedeutung der Konstanten können Sie bei mysql_fetch_array() nachlesen.

Unter Berücksichtigung der Ausführungsgeschwindigkeit ist diese Funktion identisch zu mysql_fetch_array() und fast so schnell wie mysql_fetch_row() (der Unterschied ist unwesentlich).

Beispiel 1. mysql_fetch_object() Beispiel

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db($db);
$result = mysql_query("select * from table");
while ($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

Siehe auch: mysql_fetch_array() und mysql_fetch_row().

mysql_fetch_row (PHP 3, PHP 4 >= 4.0.0)

Liefert einen Datensatz als indiziertes Array

array **mysql_fetch_row** (resource Ergebnis-Kennung)

Rückgabewert: Ein Array das der Ergebniszeile entspricht oder `FALSE` wenn keine weiteren Zeilen verfügbar sind.

mysql_fetch_row() liefert eine Datensatz aus dem Anfrageergebnis mit der übergebenen Kennung. Der Datensatz wird als Array geliefert. Jedes Feld wird in einem Array-Offset abgelegt, der Offset beginnt bei 0.

Wiederholtes Aufrufen von **mysql_fetch_row()** liefert den nächsten Datensatz des Anfrageergebnisses oder `FALSE`, wenn keine weiteren Datensätze verfügbar sind.

Siehe auch: mysql_fetch_array(), mysql_fetch_object(), mysql_data_seek(), mysql_fetch_lengths() und mysql_result().

mysql_field_flags (PHP 3, PHP 4 >= 4.0.0)

Liefert die Flags eines Feldes in einem Anfrageergebnis

string **mysql_field_flags** (resource Ergebnis, int Feldoffset)

mysql_field_flags() liefert die Flags des Feldes mit dem übergebenen Offset. Die Flags werden als einzelne Worte pro Flag und durch ein einziges Leerzeichen getrennt geliefert, so dass sie leicht mit explode() getrennt werden können.

Die folgenden Flags werden geliefert, wenn Ihre Version von MySQL diese unterstützt: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

Für Abwärtskompatibilität kann **mysql_fieldflags()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_field_name (PHP 3, PHP 4 >= 4.0.0)

Liefert den Namen eines Feldes in einem Ergebnis

string **mysql_field_name** (resource Ergebnis-Kennung, int Feldindex)

mysql_field_name() liefert den Namen des Feldes, der dem angegeben Feldindex entspricht. Der Paramater *`Ergebnis-Kennung`* muss eine gültige Ergebnis-Kennung sein und *`Feldindex`* bestimmt den numerischen Offset des Feldes.

Anmerkung: Der *`Feldindex`* beginnt bei 0.

Zur Verdeutlichung: Der Index des dritten Feldes ist demnach 2, der Index des vierten Feldes 3 und so weiter.

Beispiel 1. mysql_field_name() Beispiel

```
// Die Tabelle users enthält drei Felder:
//   user_id
//   username
//   password.

$res = mysql_db_query("users", "select * from users", $link);

echo mysql_field_name($res, 0) . "\n";
echo mysql_field_name($res, 2);
```

Das oben angeführte Beispiel liefert die folgende Ausgabe:

```
user_id
password
```

Für Abwärtskompatibilität kann **mysql_fieldname()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_field_len (PHP 3, PHP 4 >= 4.0.0)

Liefert die Länge des angegebenen Feldes

int **mysql_field_len** (resource Ergebnis, int Feldoffset)

mysql_field_len() liefert die Länge des angegebenen Feldes.

Für Abwärtskompatibilität kann **mysql_fieldlen()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_field_seek (PHP 3, PHP 4 >= 4.0.0)

Setzt den Ergebniszeiger auf ein bestimmtes Feldoffset

int **mysql_field_seek** (resource Ergebnis, int Feldoffset)

Setzt den Feldzeiger auf den angegeben Feldoffset. Wird beim nächsten Aufruf von mysql_fetch_field() kein Feldoffset übergeben, wird der Feldoffset zurückgeliefert, der bei **mysql_field_seek()** angegeben wurde.

Siehe auch: mysql_fetch_field().

mysql_field_table (PHP 3, PHP 4 >= 4.0.0)

Liefert den Namen der Tabelle, die das genannte Feld enthält

string **mysql_field_table** (resource Ergebnis-Kennung, int Feldoffset)

Liefert den Namen der Tabelle, die das genannte Feld enthält.

Für Abwärtskompatibilität kann **mysql_fieldtable()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_field_type (PHP 3, PHP 4 >= 4.0.0)

Liefert den Typ eines Feldes in einem Ergebnis

string **mysql_field_type** (resource Ergebnis, int Feldoffset)

mysql_field_type() ist ähnlich zu mysql_field_name(). Die Argumente sind identisch, aber stattdessen wird der Typ des Feldes zurückgegeben. Dieser kann "int", "real", "string", "blob", oder ein anderer Typ sein. Eine ausführliche Beschreibung aller Typen finden Sie in der MySQL Dokumentation (http://www.mysql.com/documentation/).

Beispiel 1. Mysql-Feldtypen

```
<?php

mysql_connect ("localhost:3306");
mysql_select_db ("wisconsin");
$result = mysql_query ("SELECT * FROM onek");
$fields = mysql_num_fields ($result);
$rows   = mysql_num_rows ($result);
$i = 0;
$table = mysql_field_table ($result, $i);
echo "Ihre '".$table."' Tabelle hat ".$fields." Felder und ".$rows." Daten-
sätze <BR>";
echo "Die Tabelle enthält folgende Felder <BR>";
while ($i < $fields) {
    $type  = mysql_field_type  ($result, $i);
    $name  = mysql_field_name  ($result, $i);
    $len   = mysql_field_len   ($result, $i);
    $flags = mysql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
mysql_free_result($result);
mysql_close();

?>
```

Für Abwärtskompatibilität kann **mysql_fieldtype()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_free_result (PHP 3, PHP 4 >= 4.0.0)

Gibt belegten Speicher wieder frei

bool **mysql_free_result** (resource Ergebnis-Kennung)

mysql_free_result() gibt den Speicher frei, der mit der *`Ergebnis-Kennung`* assoziert ist.

Die Funktion muss nur dann aufgerufen werden, wenn Sie sich bei Anfragen, die große Ergebnismengen liefern, Sorgen über den Speicherverbrauch zur Laufzeit des PHP-Skripts machen. Nach Ablauf des PHP-Skripts wird der Speicher ohnehin freigegeben.

Für Abwärtskompatibilität kann **mysql_freeresult()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_insert_id (PHP 3, PHP 4 >= 4.0.0)

Liefert die ID einer vorherigen INSERT-Operation

int **mysql_insert_id** ([resource Verbindungs-Kennung])

mysql_insert_id() liefert die ID, die bei der letzten INSERT-Operation für ein Feld vom Typ AUTO_INCREMENT vergeben wurde. Wenn die *`Verbindungs-Kennung`* nicht angegeben wird, wird die zuletzt geöffnete Verbindung angenommen.

mysql_insert_id() liefert 0, wenn die vorhergehende Abfrage keinen AUTO_INCREMENT Wert erzeugt hat. Falls Sie den Wert zur späteren Verwendung speichern möchten, stellen Sie sicher, dass Sie **mysql_insert_id()** direkt nach der Abfrage aufrufen, die einen Wert erzeugt hat.

Anmerkung: Der Wert der MySQL SQL Funktion `LAST_INSERT_ID()` liefert immer den zuletzt erzeugten AUTO_INCREMENT Wert. Dieser wird zwischen Abfragen nicht zurückgesetzt.

Warnung

mysql_insert_id() konvertiert den Typ der Rückgabe der nativen MySQL C API Funktion `mysql_insert_id()` in den Typ `long` (als int in PHP bezeichnet). Falls Ihre AUTO_INCREMENT Spalte vom Typ BIGINT ist, ist der Wert den **mysql_insert_id()** liefert, nicht korrekt. Verwenden Sie in diesem Fall stattdessen die MySQL interne SQL Funktion `LAST_INSERT_ID()` in einer Abfrage.

mysql_list_dbs (PHP 3, PHP 4 >= 4.0.0)

Auflistung der verfügbaren Datenbanken auf einem MySQL Server

resource **mysql_list_dbs** ([resource Verbindungs-Kennung])

mysql_list_dbs() liefert eine Ergebnis-Kennung, die alle Datenbanken auf dem Datenbankserver enthält. Um diese Ergebnis-Kennung zu durchlaufen benutzen Sie die Funktion mysql_tablename(), oder irgendeine der Funktionen, die mit Ergebnistabellen umgehen kann.

Beispiel 1. mysql_list_dbs() Beispiel

```
<?php
$link = mysql_connect('localhost', 'myname', 'secret');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
?>
```

Das oben gezeigte Beispiel liefert die folgende Ausgabe:

```
database1
database2
database3
```

Anmerkung: Dieser Code funktioniert genauso leicht mit mysql_fetch_row() oder anderen ähnlichen Funktionen.

Für Abwärtskompatibilität kann **mysql_listdbs()** verwendet werden. Diese Funktion ist jedoch veraltet.

Siehe auch: mysql_db_name().

mysql_list_fields (PHP 3, PHP 4 >= 4.0.0)

Listet MySQL Ergebnisfelder auf

resource **mysql_list_fields** (string Datenbankname, string Tabellenname [, resource Verbindungs-Kennung])

mysql_list_fields() fordert Information über eine Tabelle an. Argumente sind der Datenbankname und der Tabellenname. Die zurückgelieferte Ergebnis-Kennung kann von den Funktionen mysql_field_flags(), mysql_field_len(), mysql_field_name() und mysql_field_type() verwendet werden.

Beispiel 1. mysql_list_fields() Beispiel

```
<?php
$link = mysql_connect('localhost', 'myname', 'secret');

$fields = mysql_list_fields("database1", "table1", $link);
$columns = mysql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo mysql_field_name($fields, $i) . "\n";;
}
```

Das oben gezeigte Beispiel liefert die folgende Ausgabe:

```
field1
field2
field3
...
```

Für Abwärtskompatibilität kann **mysql_listfields()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_list_tables (PHP 3, PHP 4 >= 4.0.0)

Listet Tabellen in einer MySQL Datenbank auf

resource **mysql_list_tables** (string Datenbankname [, resource Verbindungs-Kennung])

mysql_list_tables() benötigt die Angabe eines Datenbanknamens und liefert eine Ergebniskennung ähnlich wie die Funktion mysql_db_query(). Um den aktuellen Tabellennamen aus der Ergebniskennung zu erhalten, können Sie die Funktion mysql_tablename() benutzen, oder irgendeine andere Funktion, die mit Ergebnistabellen umgehen kann.

Für Abwärtskompatibilität kann **mysql_listtables()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_num_fields (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Felder in einem Ergebnis

int **mysql_num_fields** (resource Ergebnis-Kennung)

mysql_num_fields() liefert die Anzahl der Felder in einem Ergebnis.

Siehe auch: mysql_query(), mysql_fetch_field() und mysql_num_rows().

Für Abwärtskompatibilität kann **mysql_numfields()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_num_rows (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Datensätze im Ergebnis

int **mysql_num_rows** (resource Ergebnis-Kennung)

mysql_num_rows() liefert die Anzahl der Datensätze einer Ergebnismenge. Diese Funktion ist nur gültig für SELECT Befehle. Haben Sie eine INSERT, UPDATE oder DELETE Abfrage ausgeführt und möchten die Anzahl der betroffenen Datensätze ermitteln, verwenden Sie die Funktion mysql_affected_rows().

Beispiel 1. mysql_num_rows() Beispiel

```
<?php

$link = mysql_connect("localhost", "username", "password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

Siehe auch: mysql_affected_rows(), mysql_connect(), mysql_select_db() und mysql_query().

Für Abwärtskompatibilität kann **mysql_numrows()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_pconnect (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine persistente Verbindung zum MySQL Server

resource **mysql_pconnect** ([string Server [, string Benutzername [, string Benutzerkennwort]]])

Rückgabewert: Eine positive, MySQL-persistente Verbindungs-Kennung im Erfolgsfall oder `FALSE` im Fehlerfall.

mysql_pconnect() baut eine Verbindung zu einem MySQL Server auf. Für fehlende optionale Parameter werden folgende Standards angenommen: *`Server`* = 'localhost:3306', *`Benutzername`*

= Name des Benutzers dem der Server-Prozess gehört und `Benutzerkennwort` = leeres Kennwort.

Der Parameter `Server` kann ausserdem eine Portnummer enthalten, z.B. "hostname:port" oder den Pfad zu einem Socket z.B. ":/pfad/zu/socket" für Zugriffe auf dem lokalen Rechner.

Anmerkung: Unterstützung für ":port" wurde in PHP 3.0B4 hinzugefügt.

Unterstützung für ":/pfad/zum/socket" wurde in PHP 3.0.10 hinzugefügt.

mysql_pconnect() verhält sich sehr ähnlich zu mysql_connect(), weist aber zwei wesentliche Unterschiede auf.

Erstens: vor dem Verbindungsaufbau wird zunächst versucht eine offene (persistente) Verbindung zum gleichen Host, mit dem gleichen Benutzernamen und Benutzerkennwort zu finden. Wenn das gelingt, wird eine Verbindungskennung dieser Verbindung zurückgeliefert anstatt eine neue Verbindung aufzubauen.

Zweitens: die Verbindung zum SQL Server wird beim Beenden des PHP-Skripts nicht geschlossen. Sie bleibt zur zukünftigen Verwendung bestehen. (mysql_close() schließt keine von **mysql_pconnect()** geöffneten Verbindungen).

Deshalb wird dieser Verbindungstyp 'persistent' genannt.

Anmerkung: Beachten Sie, dass diese Art der Verbindung nur funktioniert. wenn Sie PHP als Modul-Version benutzen. Für mehr Informationen schauen Sie unter Persistente Datenbankverbindungen nach.

Warnung

Die Verwendung von persistenten Verbindungen verlangt unter Umständen eine feinere Abstimmung der Konfiguration von Apache und MySQL. Dadurch sollten Sie sicherstellen, dass Sie die Anzahl der Verbindungen, die MySQL maximal erlaubt, nicht überschreiten.

mysql_query (PHP 3, PHP 4 >= 4.0.0)

Sendet eine Anfrage an MySQL

resource **mysql_query** (string Anfrage [, resource Verbindungs-Kennung])

mysql_query() sendet eine Anfrage an die zur Zeit aktiven Datenbank, die mit der angegebenen `Verbindungs-Kennung` assoziiert wird. Wird die `Verbindungs-Kennung` nicht angegeben, wird die zuletzt geöffnete Verbindung angenommen. Wenn keine offene Verbindung besteht, wird versucht eine Verbindung aufzubauen, wie beim Aufruf von mysql_connect() ohne Argumente, und diese wird dann benutzt.

Anmerkung: Die Anfrage sollte nicht mit einem Semikolon enden.

Nur für SELECT,EXPAIN, SHOW oder DESCRIBE Anweisungen liefert **mysql_query()** eine Ressourcen-Kennung oder `FALSE`, falls die Anfrage nicht korrekt ausgeführt wurde. Für alle anderen SQL Anweisungen ist der Rückgabewert von **mysql_query()** im Erfolgsfall `TRUE`, im Fehlerfall `FALSE`. Ein Rückgabewert ungleich `FALSE` bedeutet, dass die Anfrage gültig war und vom Server ausgeführt werden konnte. Das alleine sagt jedoch überhaupt nichts über die Anzahl veränderter oder gelieferter Datensätze aus. Es ist durchaus möglich, dass eine Anfrage erfolgreich war, ohne einen einzigen Datensatz zur verändern oder zu liefern.

Die folgende Anfrage ist syntaktisch falsch, deshalb liefert **mysql_query()** `FALSE` zurück:

Beispiel 1. mysql_query()

```
<php
$result = mysql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

Die folgende Anfrage ist semantisch falsch, wenn `my_col` kein Feld in der Tabelle `my_tbl` ist. Deshalb schlägt **mysql_query()** fehl und liefert `FALSE`:

Beispiel 2. mysql_query()

```
<?php
$result = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() schlägt ebenso fehl und liefert `FALSE`, wenn Sie keine Berechtigung zum Zugriff auf die verwiesene(n) Tabelle(n) haben.

Angenommen eine Anfrage ist erfolgreich, dann können Sie für eine SELECT Anweisung mit mysql_num_rows() die Anzahl der gelieferten Datensätze herausfinden. Falls Sie DELETE, INSERT, REPLACE, oder UPDATE benutzt haben, können Sie die Anzahl der betroffenen Datensätze mit mysql_affected_rows() feststellen.

Nur bei SELECT, EXPLAIN, SHOW oder DESCRIBE Anweisungen liefert **mysql_query()** eine neue Ergebnis-Kennung, die Sie an die Funktion mysql_fetch_array() und andere Funktionen, die mit Ergebnistabellen umgehen können, übergeben können. Sind sie mit der Bearbeitung Ihrer Ergebnisse fertig, können Sie die belegten Resourcen durch den Aufruf von mysql_free_result() freigeben. Ist die Ausführung des Skripts abgeschlossen, wird der belegte Speicher automatisch freigegeben.

Siehe auch: mysql_num_rows() mysql_affected_rows(), mysql_unbuffered_query(), mysql_free_result(), mysql_fetch_array(), mysql_fetch_row(), mysql_fetch_assoc(), mysql_result(), mysql_select_db() und mysql_connect().

mysql_unbuffered_query (PHP 4 >= 4.0.6)

Sendet eine SQL Anfrage an MySQL, ohne Ergebniszeilen abzuholen und zu puffern.

resource **mysql_unbuffered_query** (string Anfrage [, resource Verbindungs-Kennung])

mysql_unbuffered_query() sendet eine SQL `Anfrage` an MySQL, ohne dass die Datensätze des Ergebnisses automatisch geholt und gepuffert werden, wie es bei der Verwendung von mysql_query() der Fall ist. Einerseits spart dieses Vorgehen eine erhebliche Menge an Speicher bei SQL Anfragen ein, die große Ergebnismengen liefern. Andererseits können Sie unmittelbar mit dem Anfrageergebnis arbeiten, sobald der erste Datensatz gefunden wurde: Sie müssen nicht darauf warten, bis die gesamte SQL Anfrage abgeschlossen ist. Wird auf verschiedene Datenbanken gleichzeitig zugegriffen, muss der optionale Parameter `Verbindungs-Kennung` angegeben werden.

Anmerkung: Die Vorteile von **mysql_unbuffered_query()** haben ihren Preis: Auf ein Ergebnis von **mysql_unbuffered_query()** kann die Funktion mysql_num_rows() nicht angewendet werden. Ausserdem müssen Sie sicherstellen, dass Sie alle Datensätze des Ergebnisses einer ungepufferteten SQL Anfrage abgeholt haben. Erst dann können Sie eine neue SQL Anfrage an MySQL absetzen.

Siehe auch: mysql_query().

mysql_result (PHP 3, PHP 4 >= 4.0.0)

Liefert Ergebnis

mixed **mysql_result** (resource Ergebnis-Kennung, int Datensatz [, mixed Feld])

mysql_result() liefert den Inhalt eines Felds aus einem Anfrageergebnis. Das Argument `Feld` kann der Feldname, der Feldoffset ein Bezeichner in der Form 'Tabellenname.Feldname.' sein. Wenn das Feld einen Alias besitzt ('select foo as bar from...') muss der Alias anstatt des Feldnamens verwendet werden.

Wenn auf Anfrageergebnisse mit vielen Datensätzen zugegriffen werden soll, sollten Sie Funktionen, die auf ganze Datensätze zugreifen,in Betracht ziehen (siehe unten). Diese Funktionen liefern bei einem einzigen Aufruf den Inhalt mehrerer Felder und sind aus diesem Grund SEHR viel schneller als **mysql_result()**. Beachten Sie auch, dass die Angabe eines numerischen Offsets für ein Feld sehr viel schneller ist als die übrigen möglichen Angaben.

Aufrufe von **mysql_result()** sollten nicht mit Aufrufen anderer Funktionen verschachtelt werden, die auch auf das Ergebnis zugreifen.

Empfohlende, leistungsfähigere Alternativen: mysql_fetch_row(), mysql_fetch_array() und mysql_fetch_object().

mysql_select_db (PHP 3, PHP 4 >= 4.0.0)

Auswahl einer Datenbank

bool **mysql_select_db** (string Datenbankname [, resource Verbindungs-Kennung])

Rückgabewert: Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

mysql_select_db() aktiviert die Datenbank auf dem Server zu dem die Verbindung mit der übergegebenen Verbindungs-Kennung besteht. Fehlt die Verbindungs-Kennung wird die zuletzt geöffnete Verbindung verwendet. Wenn keine Verbindung existiert, wird versucht eine Verbindung aufzubauen, wie beim Aufruf von mysql_connect() ohne Angabe von Argumenten.

Jeder weitere Aufruf von mysql_query() benutzt diese aktive Datenbank.

Siehe auch: mysql_connect(), mysql_pconnect() und mysql_query().

Für Abwärtskompatibilität kann **mysql_selectdb()** verwendet werden. Diese Funktion ist jedoch veraltet.

mysql_tablename (PHP 3, PHP 4 >= 4.0.0)

Liefert den Namen einer Tabelle

string **mysql_tablename** (resource Ergebnis-Kennung, int i)

mysql_tablename() ermittelt aus einer Ergebnis-Kennung, die von mysql_list_tables() geliefert wurde, sowie einem Index vom Typ integer, den Namen der Tabelle. Die Funktion mysql_num_rows() kann verwendet werden, um die Anzahl der Tabellen in der Ergebnis-Kennung zu ermitteln.

Beispiel 1. mysql_tablename() Bespiel

```
<?php
mysql_connect("host");
$result = mysql_list_tables("wisconsin");
for ($i = 0; $i < mysql_num_rows($result); $i++) {
    $tb_names[$i] = mysql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
}
?>
?>
```

mysql_get_client_info (PHP 4 >= 4.0.5)

Liefert MySQL Clientinformationen

string **mysql_get_client_info** (void)

mysql_get_client_info() liefert eine Zeichenkette mit der Version der Client-Bibliothek.

mysql_get_client_info() wurde in PHP 4.0.5 hinzugefügt.

mysql_get_host_info (PHP 4 >= 4.0.5)

Liefert MySQL Host Informationen

string **mysql_get_host_info** ([resource Verbindungs-Kennung])

mysql_get_host_info() liefert eine Zeichenkette, die den Typ der benutzten Verbindung *Verbindungs-Kennung* beschreibt. Der Hostname des Servers ist ebenfalls enthalten. Fehlt der Parameter *Verbindungs-Kennung*, wird die zuletzt hergestellte Verbindung benutzt.

mysql_get_host_info() wurde in PHP 4.0.5 hinzugefügt.

mysql_get_proto_info (PHP 4 >= 4.0.5)

Liefert MySQL Protokollinformationen

int **mysql_get_proto_info** ([resource Verbindungs-Kennung])

mysql_get_proto_info() liefert die Protokollversion, die die Vebindung *Verbindungs-Kennung* nutzt. Fehlt der Parameter *Verbindungs-Kennung*, wird die zuletzt hergestellte Verbindung benutzt.

mysql_get_proto_info() wurde in PHP 4.0.5 hinzugefügt.

mysql_get_server_info (PHP 4 >= 4.0.5)

Liefert MySQL Server Informationen

int **mysql_get_server_info** ([resource Verbindungs-Kennung])

mysql_get_server_info() liefert die Serverversion, die die Vebindung mit der *Verbindungs-Kennung* nutzt. Fehlt die *Verbindungs-Kennung*, wird die zuletzt hergestellte Verbindung benutzt.

mysql_get_server_info() wurde in PHP 4.0.5 hinzugefügt.

LXII. Mohawk Software Session Handler Funktionen

msession ist eine Schnittstelle zu einem Hochgeschwindigkeitsdaemon, der entweder lokal oder auf einem entfernten Rechner (remote) laufen kann. Sie wurde erstellt um ein konsitentes Session-Management für die PHP-Webseiten zur Verfügung zu stellen.

Die Session Serversoftware finden Sie unter http://www.mohawksoft.com/phoenix.html.

msession_connect (PHP 4 CVS only)

Baut eine Verbindung zum msession-Server auf

bool **msession_connect** (string host, string port)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_disconnect (PHP 4 CVS only)

Schließt eine Verbindung zum msession-Server

void **msession_disconnect** (void)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_count (PHP 4 CVS only)

Get session count

int **msession_count** (void)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_create (PHP 4 CVS only)

Erstellt eine Session

bool **msession_create** (string session)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_destroy (PHP 4 CVS only)

Zerstört eine Session

bool **msession_destroy** (string name)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_lock (PHP 4 CVS only)

Sperrt eine Session

int **msession_lock** (string name)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_unlock (PHP 4 CVS only)

Entsperrt eine Session

int **msession_unlock** (string session, int key)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_set (PHP 4 CVS only)

Setzt einen Wert in der Session

bool **msession_set** (string session, string name, string value)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_get (PHP 4 CVS only)

Gibt einen Wert aus der Session zurück

string **msession_get** (string session, string name, string value)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_uniq (PHP 4 CVS only)

Gibt eine eindeutige ID zurück

string **msession_uniq** (int param)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_randstr (PHP 4 CVS only)

Gibt eine Zufallszeichenkette zurück

string **msession_randstr** (int param)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_find (PHP 4 CVS only)

Sucht nach einen Wert

array **msession_find** (string name, string value)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_list (PHP 4 CVS only)

List ... ?

array **msession_list** (void)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_get_array (PHP 4 CVS only)

Gibt ein Array mit ... ? zurück

array **msession_get_array** (string session)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_set_array (PHP 4 CVS only)

Setzt ein Array mit ...?

bool **msession_set_array** (string session, array tuples)

> **Warnung**
>
> Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_listvar (PHP 4 CVS only)

Listet Sessions mit Variablen auf

array **msession_listvar** (string name)

Gibt ein assoziatives Array [Wert, Session] für alle Sessions mit einer Variablen namens `name` zurück.

Wird verwendet, um Sessions mit allgemeinen Attributen zu durchsuchen.

msession_timeout (PHP 4 CVS only)

Setzt/liefert Session Timeout

int **msession_timeout** (string session [, int param])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_inc (PHP 4 CVS only)

Erhöht einen Wert innerhalb einer Session

string **msession_inc** (string session, string name)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_getdata (unknown)

Get data ... ?

string **msession_getdata** (string session)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_setdata (unknown)

Set data ... ?

bool **msession_setdata** (string session, string value)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

msession_plugin (PHP 4 CVS only)

Ruft eine Escape-Funktion innerhalb des msession peronality plugins auf.

string **msession_plugin** (string session, string val [, string param])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LXIII. muscat functions

muscat_setup (PHP 4 >= 4.0.5)

Creates a new muscat session and returns the handle. Size is the ammount of memory in bytes to allocate for muscat muscat_dir is the muscat installation dir e.g. "/usr/local/empower", it defaults to the compile time muscat directory

resource **muscat_setup** (int size [, string muscat_dir])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

muscat_setup_net (PHP 4 >= 4.0.5)

Creates a new muscat session and returns the handle. muscat_host is the hostname to connect to port is the port number to connect to - actually takes exactly the same args as fsockopen

resource **muscat_setup_net** (string muscat_host, int port)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

muscat_give (PHP 4 >= 4.0.5)

Sends string to the core muscat api

int **muscat_give** (resource muscat_handle, string string)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

muscat_get (PHP 4 >= 4.0.5)

Gets a line back from the core muscat api. Returns a literal FALSE when there is no more to get (as opposed to ""). Use === FALSE or !== FALSE to check for this

string **muscat_get** (resource muscat_handle)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

muscat_close (PHP 4 >= 4.0.5)

Shuts down the muscat session and releases any memory back to php. [Not back to the system, note!]

int **muscat_close** (resource muscat_handle)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LXIV. Netzwerk Funktionen

checkdnsrr (PHP 3, PHP 4 >= 4.0.0)

Prüft DNS-Einträge auf Übereinstimmung mit einem gegebenen Internet-Host-Namen oder einer IP Adresse

int **checkdnsrr** (string host [, string type])

Durchsucht den DNS (Domain-Name-Service) auf Daten des Typs `type` auf Übereinstimmung mit `host`. Gibt `TRUE` / wahr zurück, wenn ein Datensatz gefunden wurde. Ansonsten - oder wenn ein Fehler vorliegt - wird `FALSE` / falsch zurück gegeben.

Der Parameter `type` kann sein: A, MX, NS, SOA, PTR, CNAME, or ANY. Die Vorgabe ist MX.

Der Parameter `host` kann eine durch Punkte getrennte IP-Adresse (z.B. 192.168.0.2) oder ein Host-Name (z.B. tux.meinsein.de) sein.

Siehe auch getmxrr(), gethostbyaddr(), gethostbyname(), gethostbynamel() und das named(8) Manual.

closelog (PHP 3, PHP 4 >= 4.0.0)

Schließt die Verbindung zum System-Logger

int **closelog** (void)

closelog() schließt den Descriptor, der benötigt wird um in den System-Logger zu schreiben. Der Gebrauch von **closelog()** ist optional.

Siehe auch define_syslog_variables(), syslog() und openlog().

debugger_off (PHP 3)

Schaltet den internen PHP-Debugger aus

int **debugger_off** (void)

Schaltet den internen PHP-Debugger aus. Der Debugger befindet sich noch in der Entwicklung.

debugger_on (PHP 3)

Schaltet den internen PHP-Debugger ein

int **debugger_on** (string address)

Schaltet den internen PHP-Debugger, zeigend auf `address`, ein. Der Debugger befindet sich noch in der Entwicklung.

define_syslog_variables (PHP 3, PHP 4 >= 4.0.0)

Initialisiert alle SysLog bezogenen Konstanten

void **define_syslog_varaibles** (void)

Initialisiert alle Konstanten, die in den SysLog-Funktionen Verwendung finden.

Siehe auch openlog(), syslog() und closelog().

fsockopen (PHP 3, PHP 4 >= 4.0.0)

Stellt eine Internet- oder Unix-Domain-Socket-Verbindung her

int **fsockopen** ([string udp://hostname, int port [, int errno [, string errstr [, double timeout]]]])

Öffnet eine Socket-Verbindung zum Internet (AF_INET, unter Verwendung von TCP oder UDP) oder unter Unix (AF_UNIX). Für eine Internet-Verbindung wird eine TCP-Socket-Verbindung zu *hostname* über den Port *port* hergestellt. Für eine UDP-Verbindung müssen Sie das Protokoll *udp://hostname* explizit angeben. Bei einer UNIX-Verbindung wird *hostname* als Socket-Pfad interpretiert, *port* muss in diesem Fall 0 sein. Der optionale Parameter *timeout* kann benutzt werden, um eine Fehlerwartezeit (Timeout-Wert) für den Verbindungsaufbau (in Sekunden) zu setzen.

Die Funktion **fsockopen()** gibt einen Zeiger auf eine Datei zurück, der in Verbindung mit anderen Datei-Funktionen (wie z.B. fgets(), fgetss(), fputs(), fclose(), feof()) gebraucht werden kann.

Wenn der Aufruf der Funktion scheitert, gibt er `FALSE` / falsch zurück und wenn die optionalen *errno* und *errstr* Argumente vorhanden sind, enthalten sie den aktuellen System-Fehler, der beim Aufruf der System-Funktion `connect()` (Verbindungsaufbau) erscheint bzw. passiert ist. Wenn die erhaltene errno 0 ist und die Funktion `FALSE`/falsch zurück gibt, ist dies ein Hinweis, dass der Fehler vor dem `>connect()`-Aufruf geschah. Das ist ein Problem, dass häufig mit der Initialisierung des Sockets zusammen hängt. Beachten Sie bitte, dass das *errno* und *errstr*-Argument als Referenz zu übergeben sind.

Abhängig von der vorhandenen Umgebung kann die Verbindung zu einer UNIX-Domäne oder der optionale Verbindungsaufbau-Zeitfehler nicht zur Verfügung stehen.

Voreingestellt ist der Verbindungsaufbau im Block-Modus. Sie können dies auf Nicht-Block-Modus ändern, indem Sie socket_set_blocking() benutzen.

Beispiel 1. fsockopen()-Beispiel:

```
$fp = fsockopen ("www.php.net", 80, &$errno, &$errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br>\n";
} else {
    fputs ($fp, "GET / HTTP/1.0\r\n\r\n");
    while (!feof($fp)) {
        echo fgets($fp,128);
    }
    fclose($fp);
}
```

Das folgende Beispiel zeigt, wie das Datum und die Zeit des UDP-Service "daytime" (Port 13) von ihrer eigenen Maschine empfangen werden kann:

Beispiel 2. Gebrauch einer UDP-Verbindung:

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, &$errno; &$errstr);
if (!$fp) {
    echo "Fehler: $errno - $errstr<br>\n";
} else {
    fwrite($fp,"\n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

Siehe auch: pfsockopen(), socket_set_blocking(), socket_set_timeout(), fgets(), fgetss(), fputs(), fclose() und feof().

gethostbyaddr (PHP 3, PHP 4 >= 4.0.0)

Ermittelt den Internet-Host-Namen (z.B. tux.meinsein.de) passend zur angegebenen IP-Adresse (z.B. 192.168.0.2)

string **gethostbyaddr** (string ip_address)

Gibt den Internet-Host-Namen passend zu *ip_address* zurück. Bei einem Fehler wird die *ip_address* zurück gegeben.

Siehe auch: gethostbyname().

gethostbyname (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die IP-Adresse (z.B. 192.168.0.2) passend zum angegebenen Internet-Host-Namen (z.B. tux.meinsein.de)

string **gethostbyname** (string hostname)

Gibt die IP-Adresse passend zu *hostname* zurück.

Siehe auch: gethostbyaddr().

gethostbynamel (PHP 3, PHP 4 >= 4.0.0)

Ermittelt eine Liste von IP-Adressen passend zum angegebenen Internet-Host-Namen

array **gethostbynamel** (string hostname)

Gibt eine Liste von IP-Adressen zurück, zu denen der angegebene Internet-Host *hostname* aufgelöst wurde.

Siehe auch gethostbyname(), gethostbyaddr(), checkdnsrr(), getmxrr(), und die `named(8)` Bedienungs-Anleitung.

getmxrr (PHP 3, PHP 4 >= 4.0.0)

Ermittelt die (DNS) MX-Datensätze passend zu einem angegebenen Internet-Host-Namen

int **getmxrr** (string hostname, array mxhosts [, array weight])

Sucht im DNS nach MX-Datensätzen die zu *hostname* passen. Gibt `TRUE`/wahr zurück, wenn irgendwelche Daten gefunden wurden, ansonsten - oder wenn ein Fehler auftrat - wird `FALSE` / falsch zurück gegeben.

Eine Liste der gefundenen MX-Einträge wird im Array *mxhosts* zurück gegeben. Wenn das optionale *weight*-Array vorhanden ist, wird es mit den gefundenen Prioritäts-Informationen gefüllt.

See also checkdnsrr(), gethostbyname(), gethostbynamel(), gethostbyaddr(), und die `named(8)` Bedienungs-Anleitung.

getprotobyname (PHP 4 >= 4.0.0)

Ermittelt die Protokoll-Nummer anhand des Protokoll-Namens

int **getprotobyname** (string name)

Die Funktion **getprotobyname()** gibt die Protokoll-Nummer passend zu dem Protokoll *name* (z.B. TCP, IP, UDP, RAW) anhand der Datei `/etc/protocols` zurück.

Siehe auch getprotobynumber().

getprotobynumber (PHP 4 >= 4.0.0)

Ermittelt den Protokoll-Namen anhand der Protokoll-Nummer

string **getprotobynumber** (int number)

Die Funktion **getprotobynumber()** gibt den Protokoll-Namen passend zur angegebenen Protokoll-*number* anhand der Datei `/etc/protocols` zurück.

Siehe auch: getprotobyname().

getservbyname (PHP 4 >= 4.0.0)

Ermittelt einen Port-Wert passend zu einem Internet-Dienst und Protokoll

int **getservbyname** (string service, string protocol)

Die Funktion **getservbyname()** gibt den Internet-Port zurück, der zu *service* passt. Dies geschieht anhand des angegebenen Wertes für *protocol* mittels der Datei `/etc/services`. Der Inhalt von *protocol* ist entweder `TCP` oder `UDP`.

Siehe auch: getservbyport().

getservbyport (PHP 4 >= 4.0.0)

Ermittelt einen Internet-Dienst passend zu einem Port und Protokoll

string **getservbyport** (int port, string protocol)

Die Funktion **getservbyport()** gibt den Internet-Dienst zurück, der zu *port* für das angegebene *protocol* passt (anhand der Datei `/etc/services`). Der Inhalt von *protocol* ist entweder `TCP` oder `UDP`.

Siehe auch: getservbyname().

ip2long (PHP 4 >= 4.0.0)

Verwandelt eine gemäß IP-V4-Protokoll angegebene Internet-Adresse vom Punkt-Format in die ausgeschriebene Adress-Angabe

int **ip2long** (string ip_address)

Die Funktion **ip2long()** erzeugt aus einer im Punkt-Format angegebenen IP-V4-Adresse eine ausgeschriebene Adress-Angabe.

Beispiel 1. ip2long()-Beispiel:

```
<?
$ip = gethostbyname("www.php.net");
$out = "Die folgenden URLs sind gleich:<br>\n";
$out .= "http://www.php.net/, http://".$ip."/, and http://".ip2long($ip)."/<br>\n";
echo $out;
?>
```

Siehe auch: long2ip().

long2ip (PHP 4 >= 4.0.0)

Konvertiert eine IP-V4-Internet-Netzwerk-Adresse in einen String, der das Punkt-Format enthält ("Dotted-Format")

string **long2ip** (int proper_address)

Die Funktion **long2ip()** erzeugt eine Internet Adresse im Punkt-Format (also: www.xxx.yyy.zzz) anhand einer richtigen - ausgeschriebenen - Adress-Angabe.

Siehe auch: ip2long().

openlog (PHP 3, PHP 4 >= 4.0.0)

Stellt eine Verbindung zu einem Log-Dienst des Systems her

int **openlog** (string ident, int option, int facility)

Die Funktion **openlog()** kontaktiert die System-Log-Dienste eines Programms. Die Zeichenkette *`ident`* wird jeder Meldung angefügt. Werte für die Parameter *`option`* und *`facility`* werden unten erläutert. Das Argument *`option`* wird verwendet, um festzulegen, welche Log-Optionen bei der Erzeugung von Log-Nachrichten verwendet werden sollen. Das Argument *`facility`* wird verwendet, um die Art des Programmes festzulegen, das die Nachricht loggt. Dies erlaubt ihnen (in ihrer speziellen SysLog-Konfiguartion) die Festlegung der Behandlungsweise von Log-Nachrichten. Der Gebrauch von **openlog()** ist optional. Er wird - falls erforderlich - automatisch aufgerufen von syslog(). Dann ist der Vorgabewert von *`ident`* falsch / `FALSE`.

Tabelle 1. openlog() Options

Konstante	Beschreibung
LOG_CONS	Tritt ein Fehler auf beim Senden von Daten an den System-Logging-Service, wird direkt an die System-Konsole ausgegeben.
LOG_NDELAY	Öffnet sofort die Verbindung zum Log-Service.
LOG_ODELAY	(Voreinstellung); öffne die Verbindung erst, wenn die erste Meldung geloggt werden soll.
LOG_PERROR	Gib die Log-Meldungen auch an der Standard-Ausgabe für Fehler aus.
LOG_PID	Schließe die PID bei jeder Meldung ein.

Sie können eine oder mehrere dieser Optionen verwenden. Benutzen Sie mehrere, müssen diese per `OR` getrennt werden. Um die Verbindung sofort herzu stellen, auch auf die Konsole zu schreiben und die PID in jede Meldung auf zu nehmen, müssen Sie folgendes angeben: `LOG_CONS | LOG_NDELAY | LOG_PID`.

Tabelle 2. openlog()-Möglichkeiten

Konstante	Beschreibung
LOG_AUTH	Security- / Authorization-Meldungen (benutzt statt dessen LOG_AUTHPRIV in Systemen, wo diese Konstante definiert ist).
LOG_AUTHPRIV	Security- / Authorization-Meldungen (private).
LOG_CRON	Uhr-abhängige Dienste (Dämonen wie cron and at).
LOG_DAEMON	Andere System-Dienste (Dämonen).

Konstante	Beschreibung
LOG_KERN	Kernel-Meldungen.
LOG_LOCAL0 ... LOG_LOCAL7	Reserviert für lokale Benutzung.
LOG_LPR	Zeilen-Drucker Subsystem.
LOG_MAIL	Mailing-Subsystem.
LOG_NEWS	USENET-News-Subsystem.
LOG_SYSLOG	Meldungen die intern von syslogd erzeugt wurden.
LOG_USER	Generische User-Level-Meldungen.
LOG_UUCP	UUCP-Subsystem

Siehe auch define_syslog_variables(), syslog() und closelog().

pfsockopen (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Stellt eine dauernde/permanente Internet oder Unix-Domain-Socket Verbindung her

int **pfsockopen** (string hostname, int port [, int errno [, string errstr [, int timeout]]])

Diese Funktion beinhaltet das gleiche wie fsockopen() mit dem Unterschied, dass die Verbindung nicht beendet wird, sobald das Script beendet wird. Sie ist die permanente-Version von fsockopen().

socket_get_status (PHP 4 >= 4.0.0)

Gibt Informationen über eine vorhandene Socket-Ressource zurück

array **socket_get_status** (resource socket_get_status)

Gibt Informationen über eine vorhandene Socket-Ressource zurück. Z. Zt. werden vier Einträge in dem resultierenden Array vorgenommen:

- *`timed_out`* (bool) - Der Timeout-Wert des Sockets für das Warten auf Daten
- *`blocked`* (bool) - Der Socket wurde blockiert
- *`eof`* (bool) - Das EOF (End Of File) ist erreicht
- *`unread_bytes`* (int) - Anzahl der Bytes, die noch im Socket-Puffer sind

Siehe auch: accept_connect(), bind(), connect(), listen() und strerror().

socket_set_blocking (PHP 4 >= 4.0.0)

Schaltet den Block-Modus eines Sockets an oder aus

int **socket_set_blocking** (int socket descriptor, int mode)

Wenn *mode* FALSE / falsch ist, wird die angegebene Socket- Bezeichnung auf Nicht-Block-Modus gesetzt, anderenfalls auf Block-Modus. Dies beeinflusst Aufrufe von fgets(), die über diesen Sockets lesen. Im Nicht-Block-Modus wird ein fgets()-Aufruf sofort zum Script zurückkehren, während er im Block-Modus auf eingehende Daten (auf diesem Socket) warten wird.

Hinweis:: Diese Funktion hieß früher **set_socket_blocking()**, dieser Name sollte jedoch nicht länger verwendet werden.

socket_set_timeout (PHP 4 >= 4.0.0)

Setzt den Timeout-Wert eines Socket

bool **socket_set_timeout** (int socket descriptor, int seconds, int microseconds)

Setzt den Timeout-Wert von *socket descriptor*, ausgedrückt durch die Summe von *seconds* und *microseconds*.

Beispiel 1. socket_set_timeout()-Beispiel:

```
<?php
$fp = fsockopen("www.php.net", 80);
if(!$fp) {
    echo "Keine Verbindung möglich\n";
} else {
    fputs($fp,"GET / HTTP/1.0\n\n");
    $start = time();
    socket_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(socket_get_status($fp));
    fclose($fp);
    print $res;
}
?>
```

Hinweis:: Diese Funktion hieß früher **set_socket_timeout()**, dieser Name sollte jedoch nicht länger verwendet werden.

Siehe auch: fsockopen() und fopen().

syslog (PHP 3, PHP 4 >= 4.0.0)

Erzeugt eine Meldung im System-Logging

int **syslog** (int priority, string message)

Die Funktion **syslog()** erzeugt eine Status-Meldung, die an die System-Log-Funktion weitergereicht wird. Der Parameter *`priority`* ist eine Kombination der System-Möglichkeiten und dem Level. Werte hierzu werden im nächsten Abschnitt genannt. Es verbleibt die zu sendende Nachricht, in der die beiden Zeichen `%m` durch den Fehlertext (strerror) ersetzt werden, der dem Wert errno entspricht.

Tabelle 1. syslog()-Prioritäten (in absteigender Reihenfolge)

Konstante	Beschreibung:
LOG_EMERG	System ist unbrauchbar.
LOG_ALERT	Aktion ist sofort erforderlich.
LOG_CRIT	Kritischer Zustand.
LOG_ERR	Fehler aufgetreten.
LOG_WARNING	Warnung aufgetreten.
LOG_NOTICE	Normale, aber wichtige, Meldung.
LOG_INFO	Informative Meldung.
LOG_DEBUG	Debug-Level-Meldung.

Beispiel 1. Die Verwendung von syslog():

```
<?php
define_syslog_variables();
// öffne SysLog, schließe die Prozess-ID (PID) ein, sende
// die Log-Meldungen an die Standard-Fehler-Ausgabe und benutze
// einen User-definierten Log-Mechanismus.
openlog("myScripLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);

// etwas Code

if (authorized_client()) {
    // mache etwas
} else {
    // unerlaubter Client!
    // logge den Zugriff
    $access = date("Y/m/d H:i:s");
    syslog(LOG_WARNING,"Unauthorisierter Client: $access $REMOTE_ADDR ($HTTP_USER_AGENT)'
}

closelog();
?>
```

Weiter Informationen zum Benutzer-definierten Log-Handler finden Sie unter syslog.conf(5) der Unix-Manual-Seiten. Zusätzliche Informationen zu den SysLog-Möglichkeiten finden Sie in den Man-Pages zu syslog(3) (auf UNIX-Systemen).

Unter Windows-NT wird der Syslog-Dienst emuliert, indem der Ereignis-Protokollierdienst genutzt wird.

Siehe auch define_syslog_variables(), openlog() und closelog().

LXV. Ncurses terminal screen control functions

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

What is ncurses?

ncurses (new curses) is a free software emulation of curses in System V Rel 4.0 (and above). It uses terminfo format, supports pads, colors, multiple highlights, form characters and function key mapping.

Platforms

Ncurses is available for the following platforms:

- AIX
- BeOS
- Cygwin
- Digital Unix (aka OSF1)
- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

Requirements

You need the ncurses libraries and headerfiles. Download the latest version from the

ftp://ftp.gnu.org/pub/gnu/ncurses/ or from an other GNU-Mirror.

Installation

To get these functions to work, you have to compile the CGI version of PHP with `--with-ncurses`.

Ncurses predifined constants

Error codes

On error ncurses functions return NCURSES_ERR.

Colors

Tabelle 1. ncurses color constants

constant	meaning
NCURSES_COLOR_BLACK	no color (black)
NCURSES_COLOR_WHITE	white
NCURSES_COLOR_RED	red - supported when terminal is in color mode
NCURSES_COLOR_GREEN	green - supported when terminal is in color mod
NCURSES_COLOR_YELLOW	yellow - supported when terminal is in color mod
NCURSES_COLOR_BLUE	blue - supported when terminal is in color mod
NCURSES_COLOR_CYAN	cyan - supported when terminal is in color mod
NCURSES_COLOR_MAGENTA	magenta - supported when terminal is in color mod

Keys

Tabelle 2. ncurses key constants

constant	meaning
NCURSES_KEY_F0 - NCURSES_KEY_F64	function keys F1 - F64
NCURSES_KEY_DOWN	down arrow
NCURSES_KEY_UP	up arrow
NCURSES_KEY_LEFT	left arrow
NCURSES_KEY_RIGHT	right arrow
NCURSES_KEY_HOME	home key (upward+left arrow)
NCURSES_KEY_BACKSPACE	backspace
NCURSES_KEY_DL	delete line

constant	meaning
NCURSES_KEY_IL	insert line
NCURSES_KEY_DC	delete character
NCURSES_KEY_IC	insert char or enter insert mode
NCURSES_KEY_EIC	exit insert char mode
NCURSES_KEY_CLEAR	clear screen
NCURSES_KEY_EOS	clear to end of screen
NCURSES_KEY_EOL	clear to end of line
NCURSES_KEY_SF	scroll one line forward
NCURSES_KEY_SR	scroll one line backward
NCURSES_KEY_NPAGE	next page
NCURSES_KEY_PPAGE	previous page
NCURSES_KEY_STAB	set tab
NCURSES_KEY_CTAB	clear tab
NCURSES_KEY_CATAB	clear all tabs
NCURSES_KEY_SRESET	soft (partial) reset
NCURSES_KEY_RESET	reset or hard reset
NCURSES_KEY_PRINT	print
NCURSES_KEY_LL	lower left
NCURSES_KEY_A1	upper left of keypad
NCURSES_KEY_A3	upper right of keypad
NCURSES_KEY_B2	center of keypad
NCURSES_KEY_C1	lower left of keypad
NCURSES_KEY_C3	lower right of keypad
NCURSES_KEY_BTAB	back tab
NCURSES_KEY_BEG	beginning
NCURSES_KEY_CANCEL	cancel
NCURSES_KEY_CLOSE	close
NCURSES_KEY_COMMAND	cmd (command)
NCURSES_KEY_COPY	copy
NCURSES_KEY_CREATE	create
NCURSES_KEY_END	end
NCURSES_KEY_EXIT	exit
NCURSES_KEY_FIND	find
NCURSES_KEY_HELP	help
NCURSES_KEY_MARK	mark
NCURSES_KEY_MESSAGE	message
NCURSES_KEY_MOVE	move
NCURSES_KEY_NEXT	next
NCURSES_KEY_OPEN	open
NCURSES_KEY_OPTIONS	options
NCURSES_KEY_PREVIOUS	previous
NCURSES_KEY_REDO	redo

constant	meaning
NCURSES_KEY_REFERENCE	ref (reference)
NCURSES_KEY_REFRESH	refresh
NCURSES_KEY_REPLACE	replace
NCURSES_KEY_RESTART	restart
NCURSES_KEY_RESUME	resume
NCURSES_KEY_SAVE	save
NCURSES_KEY_SBEG	shiftet beg (beginning)
NCURSES_KEY_SCANCEL	shifted cancel
NCURSES_KEY_SCOMMAND	shifted command
NCURSES_KEY_SCOPY	shifted copy
NCURSES_KEY_SCREATE	shifted create
NCURSES_KEY_SDC	shifted delete char
NCURSES_KEY_SDL	shifted delete line
NCURSES_KEY_SELECT	select
NCURSES_KEY_SEND	shifted end
NCURSES_KEY_SEOL	shifted end of line
NCURSES_KEY_SEXIT	shifted exit
NCURSES_KEY_SFIND	shifted find
NCURSES_KEY_SHELP	shifted help
NCURSES_KEY_SHOME	shifted home
NCURSES_KEY_SIC	shifted input
NCURSES_KEY_SLEFT	shifted left arrow
NCURSES_KEY_SMESSAGE	shifted message
NCURSES_KEY_SMOVE	shifted move
NCURSES_KEY_SNEXT	shifted next
NCURSES_KEY_SOPTIONS	shifted options
NCURSES_KEY_SPREVIOUS	shifted previous
NCURSES_KEY_SPRINT	shifted print
NCURSES_KEY_SREDO	shifted redo
NCURSES_KEY_SREPLACE	shifted replace
NCURSES_KEY_SRIGHT	shifted right arrow
NCURSES_KEY_SRSUME	shifted resume
NCURSES_KEY_SSAVE	shifted save
NCURSES_KEY_SSUSPEND	shifted suspend
NCURSES_KEY_UNDO	undo
NCURSES_KEY_MOUSE	mouse event has occured
NCURSES_KEY_MAX	maximum key value

Mouse

Tabelle 3. mouse constants

Constant	meaning
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	button (1-4) released
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	button (1-4) pressed
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	button (1-4) clicked
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	button (1-4) double clicked
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	button (1-4) triple clicked
NCURSES_BUTTON_CTRL	ctrl pressed during click
NCURSES_BUTTON_SHIFT	shift pressed during click
NCURSES_BUTTON_ALT	alt pressed during click
NCURSES_ALL_MOUSE_EVENTS	report all mouse events
NCURSES_REPORT_MOUSE_POSITION	report mouse position

ncurses_can_change_color (PHP 4 >= 4.1.0)

Check if we can change terminals colors

bool **ncurses_can_change_color** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

The function **ncurses_can_change_color()** returns `TRUE` or `FALSE`, depending on whether the terminal has color capabilities and whether the programmer can change the colors.

ncurses_cbreak (PHP 4 >= 4.1.0)

Switch of input buffering

bool **ncurses_cbreak** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_cbreak() disables line buffering and character processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

ncurses_cbreak() returns `TRUE` or NCURSES_ERR if any error occured.

See also: ncurses_nocbreak()

ncurses_clear (PHP 4 >= 4.1.0)

Clear screen

bool **ncurses_clear** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_clear() clears the screen completely without setting blanks. Returns `FALSE` on success, otherwise `TRUE`.

Note: **ncurses_clear()** clears the screen without setting blanks, which have the current background rendition. To clear screen with blanks, use ncurses_erase().

See also: ncurses_erase()

ncurses_clrtobot (PHP 4 >= 4.1.0)

Clear screen from current position to bottom

bool **ncurses_clrtobot** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_clrtobot() erases all lines from cursor to end of screen and creates blanks. Blanks created by **ncurses_clrtobot()** have the current background rendition. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_clear(), ncurses_clrtoeol()

ncurses_clrtoeol (PHP 4 >= 4.1.0)

Clear screen from current position to end of line

bool **ncurses_clrtoeol** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_clrtoeol() erases the current line from cursor position to the end. Blanks created by **ncurses_clrtoeol()** have the current background rendition. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_clear(), ncurses_clrtobot()

ncurses_def_prog_mode (PHP 4 >= 4.1.0)

Saves terminals (program) mode

bool **ncurses_def_prog_mode** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_def_prog_mode() saves the current terminal modes for program (in curses) for use by **ncurses_reset_prog_mode()**. Returns `FALSE` on success, otherwise `TRUE`.

See also: **ncurses_reset_prog_mode()**

ncurses_def_shell_mode (PHP 4 >= 4.1.0)

Saves terminals (shell) mode

bool **ncurses_def_shell_mode** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_def_shell_mode() saves the current terminal modes for shell (not in curses) for use by **ncurses_reset_shell_mode()**. Returns `FALSE` on success, otherwise `TRUE`.

See also: **ncurses_reset_shell_mode()**

ncurses_delch (PHP 4 >= 4.1.0)

Delete character at current position, move rest of line left

bool **ncurses_delch** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_delch() deletes the character under the cursor. All characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change. Returns FALSE on success, otherwise TRUE.

See also: ncurses_deleteln()

ncurses_deleteln (PHP 4 >= 4.1.0)

Delete line at current position, move rest of screen up

bool **ncurses_deleteln** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_deleteln() deletes the current line under cursorposition. All lines below the current line are moved up one line. The bottom line of window is cleared. Cursor position does not change. Returns FALSE on success, otherwise TRUE.

See also: ncurses_delch()

ncurses_doupdate (PHP 4 >= 4.1.0)

Write all prepared refreshes to terminal

bool **ncurses_doupdate** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_doupdate()() compares the virtual screen to the physical screen and updates the physical screen. This way is more effective than using multiple refresh calls. Returns FALSE on success, TRUE if any error occured.

ncurses_echo (PHP 4 >= 4.1.0)

Activate keyboard input echo

bool **ncurses_echo** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_echo() enables echo mode. All characters typed by user are echoed by ncurses_getch(). Returns `FALSE` on success, `TRUE` if any error occured.

To disable echo mode use ncurses_noecho().

ncurses_erase (PHP 4 >= 4.1.0)

Erase terminal screen

bool **ncurses_erase** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_erase() fills the terminal screen with blanks. Created blanks have the current background rendition, set by ncurses_bkgd(). Returns `FALSE` on success, `TRUE` if any error occured.

See also: ncurses_bkgd(), ncurses_clear()

ncurses_erasechar (PHP 4 >= 4.1.0)

Returns current erase character

string **ncurses_erasechar** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_erasechar() returns the current erase char character.

See also: ncurses_killchar()

ncurses_flash (PHP 4 >= 4.1.0)

Flash terminal screen (visual bell)

bool **ncurses_flash** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_flash() flashes the screen, and if its not possible, sends an audible alert (bell). Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_beep()

ncurses_flushinp (PHP 4 >= 4.1.0)

Flush keyboard input buffer

bool **ncurses_flushinp** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

The **ncurses_flushinp()** throws away any typeahead that has been typed and has not yet been read by your program. Returns `FALSE` on success, otherwise `TRUE`.

ncurses_has_colors (PHP 4 >= 4.1.0)

Check if terminal has colors

bool **ncurses_has_colors** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_has_colors() returns `TRUE` or `FALSE` depending on whether the terminal has color capacitites.

See also: ncurses_can_change_color()

ncurses_has_ic (PHP 4 >= 4.1.0)

Check for insert- and delete-capabilities

bool **ncurses_has_ic** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_has_ic() checks terminals insert- and delete capabilitites. It returns `TRUE` when terminal has insert/delete-capabilities, otherwise `FALSE`.

See also: ncurses_has_il()

ncurses_has_il (PHP 4 >= 4.1.0)

Check for line insert- and delete-capabilities

bool **ncurses_has_il** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_has_il() checks terminals insert- and delete-line-capabilities. It returns `TRUE` when terminal has insert/delete-line capabilities, otherwise `FALSE`

See also: ncurses_has_ic()

ncurses_inch (PHP 4 >= 4.1.0)

Get character and attribute at current position

string **ncurses_inch** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_inch() returns the character from the current position.

ncurses_insertln (PHP 4 >= 4.1.0)

Insert a line, move rest of screen down

bool **ncurses_insertln** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_insertln() inserts a new line above the current line. The bottom line will be lost.

ncurses_isendwin (PHP 4 >= 4.1.0)

Ncurses is in endwin mode, normal screen output may be performed

bool **ncurses_isendwin** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_isendwin() returns `TRUE`, if **ncurses_endwin()** has been called without any subsequent calls to ncurses_wrefresh(), otherwise `FALSE`.

See also: **ncurses_endwin() ncurses_wrefresh()()**

ncurses_killchar (PHP 4 >= 4.1.0)

Returns current line kill character

bool **ncurses_killchar** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_killchar() returns the current line kill character.

See also: ncurses_erasechar()

ncurses_nl (PHP 4 >= 4.1.0)

Translate newline and carriage return / line feed

bool **ncurses_nl** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_nocbreak (PHP 4 >= 4.1.0)

Switch terminal to cooked mode

bool **ncurses_nocbreak** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_nocbreak() routine returns terminal to normal (cooked) mode. Initially the terminal may or may not in cbreak mode as the mode is inherited. Therefore a program should call ncurses_cbreak() and **ncurses_nocbreak()** explicitly. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_cbreak()

ncurses_noecho (PHP 4 >= 4.1.0)

Switch off keyboard input echo

bool **ncurses_noecho** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_noecho() prevents echoing of user typed characters. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_echo(), ncurses_getch()

ncurses_nonl (PHP 4 >= 4.1.0)

Do not translate newline and carriage return / line feed

bool **ncurses_nonl** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_noraw (PHP 4 >= 4.1.0)

Switch terminal out of raw mode

bool **ncurses_noraw** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_noraw() switches the terminal out of raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are

that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_raw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_raw (PHP 4 >= 4.1.0)

Switch terminal into raw mode

bool **ncurses_raw** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_raw() places the terminal in raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_noraw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_resetty (PHP 4 >= 4.1.0)

Restores saved terminal state

bool **ncurses_resetty** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Function **ncurses_resetty()** restores the terminal state, which was previously saved by calling ncurses_savetty(). This function always returns FALSE.

See also: ncurses_savetty()

ncurses_savetty (PHP 4 >= 4.1.0)

Saves terminal state

bool **ncurses_savetty** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Function **ncurses_savetty()** saves the current terminal state. The saved terminal state can be restored with function ncurses_resetty(). **ncurses_savetty()** always returns FALSE.

See also: ncurses_resetty()

ncurses_slk_init (PHP 4 >= 4.1.0)

Initializes soft label key functions

bool **ncurses_slk_init** (int format)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Funtion **ncurses_slk_init()** must be called before **ncurses_initscr()** or **ncurses_newterm()** is called. If **ncurses_initscr()** eventually uses a line from stdscr to emulate the soft labels, then *format* determines how the labels are arranged of the screen. Setting *format* to 0 indicates a 3-2-3 arrangement of the labels, 1 indicates a 4-4 arrangement and 2 indicates the PC like 4-4-4 mode, but in addition an index line will be created.

ncurses_slk_attr (PHP 4 >= 4.1.0)

Returns current soft label key attribute

bool **ncurses_slk_attr** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_slk_attr() returns the current soft label key attribute. On error returns TRUE, otherwise FALSE.

ncurses_slk_clear (PHP 4 >= 4.1.0)

Clears soft labels from screen

bool **ncurses_slk_clear** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

The function **ncurses_slk_clear()** clears soft label keys from screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_noutrefresh (PHP 4 >= 4.1.0)

Copies soft label keys to virtual screen

bool **ncurses_slk_noutrefresh** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_slk_refresh (PHP 4 >= 4.1.0)

Copies soft label keys to screen

bool **ncurses_slk_refresh** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_slk_refresh() copies soft label keys from virtual screen to physical screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_restore (PHP 4 >= 4.1.0)

Restores soft label keys

bool **ncurses_slk_restore** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

The function **ncurses_slk_restore()** restores the soft label keys after ncurses_slk_clear() has been performed.

ncurses_slk_touch (PHP 4 >= 4.1.0)

Fources output when ncurses_slk_noutrefresh is performed

bool **ncurses_slk_touch** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

The **ncurses_slk_touch()** function forces all the soft labels to be output the next time a ncurses_slk_noutrefresh() is performed.

ncurses_termattrs (PHP 4 >= 4.1.0)

Returns a logical OR of all attribute flags supported by terminal

bool **ncurses_termattrs** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_use_default_colors (PHP 4 >= 4.1.0)

Assign terminal default colors to color id -1

bool **ncurses_use_default_colors** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_addch (PHP 4 >= 4.1.0)

Add character at current position and advance cursor

int **ncurses_addch** (int ch)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_addchnstr (PHP 4 CVS only)

Add attributed string with specified length at current position

int **ncurses_addchnstr** (string s, int n)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_addchstr (PHP 4 CVS only)

Add attributed string at current position

int **ncurses_addchstr** (string s)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_addnstr (PHP 4 CVS only)

Add string with specified length at current position

int **ncurses_addnstr** (string s, int n)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_addstr (PHP 4 CVS only)

Output text at current position

int **ncurses_addstr** (string text)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_assume_default_colors (PHP 4 CVS only)

Define default colors for color 0

int **ncurses_assume_default_colors** (int fg, int bg)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_attroff (PHP 4 >= 4.1.0)

Turn off the given attributes

int **ncurses_attroff** (int attributes)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_attron (PHP 4 >= 4.1.0)

Turn on the given attributes

int **ncurses_attron** (int attributes)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_attrset (PHP 4 >= 4.1.0)

Set given attributes

int **ncurses_attrset** (int attributes)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_baudrate (PHP 4 >= 4.1.0)

Returns baudrate of terminal

int **ncurses_baudrate** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_beep (PHP 4 >= 4.1.0)

Let the terminal beep

int **ncurses_beep** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_beep() sends an audlible alert (bell) and if its not possible flashes the screen. Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_flash()

ncurses_bkgd (PHP 4 >= 4.1.0)

Set background property for terminal screen

int **ncurses_bkgd** (int attrchar)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_border (PHP 4 CVS only)

Draw a border around the screen using attributed characters

int **ncurses_border** (int left, int right, int top, int bottom, int tl_corner, int tr_corner, int bl_corner, int br_corner)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_color_set (PHP 4 >= 4.1.0)

Set fore- and background color

int **ncurses_color_set** (int pair)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_curs_set (PHP 4 >= 4.1.0)

Set cursor state

int **ncurses_curs_set** (int visibility)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_define_key (PHP 4 CVS only)

Define a keycode

int **ncurses_define_key** (string definition, int keycode)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_delay_output (PHP 4 >= 4.1.0)

Delay output on terminal using padding characters

int **ncurses_delay_output** (int milliseconds)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_delwin (PHP 4 >= 4.1.0)

Delete a ncurses window

int **ncurses_delwin** (resource window)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_echochar (PHP 4 >= 4.1.0)

Single character output including refresh

int **ncurses_echochar** (int character)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_end (PHP 4 >= 4.1.0)

Stop using ncurses, clean up the screen

int **ncurses_end** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_filter (PHP 4 >= 4.1.0)

int **ncurses_filter** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_getch (PHP 4 >= 4.1.0)

Read a character from keyboard

int **ncurses_getch** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_halfdelay (PHP 4 >= 4.1.0)

Put terminal into halfdelay mode

int **ncurses_halfdelay** (int tenth)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_has_key (PHP 4 >= 4.1.0)

Check for presence of a function key on terminal keyboard

int **ncurses_has_key** (int keycode)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_hline (PHP 4 CVS only)

Draw a horizontal line at current position using an attributed character and max. n characters long

int **ncurses_hline** (int charattr, int n)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_init (PHP 4 >= 4.1.0)

Initialize ncurses

int **ncurses_init** (void)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_init_color (PHP 4 CVS only)

Set new RGB value for color

int **ncurses_init_color** (int color, int r, int g, int b)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_init_pair (PHP 4 >= 4.1.0)

Allocate a color pair

int **ncurses_init_pair** (int pair, int fg, int bg)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_insch (PHP 4 >= 4.1.0)

Insert character moving rest of line including character at current position

int **ncurses_insch** (int character)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_insdelln (PHP 4 >= 4.1.0)

Insert lines before current line scrolling down (negative numbers delete and scroll up)

int **ncurses_insdelln** (int count)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_insstr (PHP 4 CVS only)

Insert string at current position, moving rest of line right

int **ncurses_insstr** (string text)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_instr (PHP 4 CVS only)

Reads string from terminal screen

int **ncurses_instr** (string buffer)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_instr() returns the number of charaters read from the current character position until end of line. `buffer` contains the characters. Atrributes are stripped from the characters.

ncurses_keyok (PHP 4 CVS only)

Enable or disable a keycode

int **ncurses_keyok** (int keycode, bool enable)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mouseinterval (PHP 4 >= 4.1.0)

Set timeout for mouse button clicks

int **ncurses_mouseinterval** (int milliseconds)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_move (PHP 4 >= 4.1.0)

Move output position

int **ncurses_move** (int y, int x)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvaddch (PHP 4 CVS only)

Move current position and add character

int **ncurses_mvaddch** (int y, int x, int c)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvaddchnstr (PHP 4 CVS only)

Move position and add attrributed string with specified length

int **ncurses_mvaddchnstr** (int y, int x, string s, int n)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvaddchstr (PHP 4 CVS only)

Move position and add attributed string

int **ncurses_mvaddchstr** (int y, int x, string s)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvaddnstr (PHP 4 CVS only)

Move position and add string with specified length

int **ncurses_mvaddnstr** (int y, int x, string s, int n)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvaddstr (PHP 4 CVS only)

Move position and add string

int **ncurses_mvaddstr** (int y, int x, string s)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvcur (PHP 4 CVS only)

Move cursor immediately

int **ncurses_mvcur** (int old_y, int old_x, int new_y, int new_x)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvdelch (PHP 4 CVS only)

Move position and delete character, shift rest of line left

int **ncurses_mvdelch** (int y, int x)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvgetch (PHP 4 CVS only)

Move position and get character at new position

int **ncurses_mvgetch** (int y, int x)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvhline (PHP 4 CVS only)

Set new position and draw a horizontal line using an attributed character and max. n characters long

int **ncurses_mvhline** (int y, int x, int attrchar, int n)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvinch (PHP 4 CVS only)

Move position and get attributed character at new position

int **ncurses_mvinch** (int y, int x)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvvline (unknown)

Set new position and draw a vertical line using an attributed character and max. n characters long

int **ncurses_mvvline** (int y, int x, int attrchar, int n)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_mvwaddstr (PHP 4 CVS only)

Add string at new position in window

int **ncurses_mvwaddstr** (resource window, int y, int x, string text)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_napms (PHP 4 >= 4.1.0)

Sleep

int **ncurses_napms** (int milliseconds)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_newwin (PHP 4 >= 4.1.0)

Create a new window

int **ncurses_newwin** (int rows, int cols, int y, int x)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_noqiflush (PHP 4 >= 4.1.0)

Do not flush on signal characters

int **ncurses_noqiflush** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_putp (PHP 4 CVS only)

int **ncurses_putp** (string text)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_qiflush (PHP 4 >= 4.1.0)

Flush on signal characters

int **ncurses_qiflush** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_refresh (PHP 4 >= 4.1.0)

Refresh screen

int **ncurses_refresh** (int ch)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_scr_dump (PHP 4 CVS only)

Dump screen content to file

int **ncurses_scr_dump** (string filename)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_scr_init (PHP 4 CVS only)

Initialize screen from file dump

int **ncurses_scr_init** (string filename)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_scr_restore (PHP 4 CVS only)

Restore screen from file dump

int **ncurses_scr_restore** (string filename)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_scr_set (PHP 4 CVS only)

Inherit screen from file dump

int **ncurses_scr_set** (string filename)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_scrl (PHP 4 >= 4.1.0)

Scroll window content up or down without changing current position

int **ncurses_scrl** (int count)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_slk_attroff (PHP 4 >= 4.1.0)

int **ncurses_slk_attroff** (int intarg)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_slk_attron (PHP 4 >= 4.1.0)

int **ncurses_slk_attron** (int intarg)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_slk_attrset (PHP 4 >= 4.1.0)

int **ncurses_slk_attrset** (int intarg)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_slk_color (PHP 4 >= 4.1.0)

Sets color for soft label keys

int **ncurses_slk_color** (int intarg)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_standend (PHP 4 >= 4.1.0)

Stop using ’standout’ attribute

int **ncurses_standend** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_standout (PHP 4 >= 4.1.0)

Start using 'standout' attribute

int **ncurses_standout** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_start_color (PHP 4 >= 4.1.0)

Start using colors

int **ncurses_start_color** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_typeahead (PHP 4 >= 4.1.0)

Specify different filedescriptor for typeahead checking

int **ncurses_typeahead** (int fd)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_ungetch (PHP 4 >= 4.1.0)

Put a character back into the input stream

int **ncurses_ungetch** (int keycode)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_use_extended_names (PHP 4 >= 4.1.0)

Control use of extended names in terminfo descriptions

int **ncurses_use_extended_names** (bool flag)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_vidattr (PHP 4 >= 4.1.0)

int **ncurses_vidattr** (int intarg)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_vline (PHP 4 CVS only)

Draw a vertical line at current position using an attributed character and max. n characters long

int **ncurses_vline** (int charattr, int n)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_wrefresh (PHP 4 CVS only)

Refresh window on terminal screen

int **ncurses_wrefresh** (resource window)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_bkgdset (PHP 4 >= 4.1.0)

Control screen background

void **ncurses_bkgdset** (int attrchar)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_timeout (PHP 4 >= 4.1.0)

Set timeout for special key sequences

void **ncurses_timeout** (int millisec)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_use_env (PHP 4 >= 4.1.0)

Control use of environment information about terminal size

void **ncurses_use_env** (bool flag)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

undocumented

ncurses_termname (PHP 4 CVS only)

Returns terminals (short)-name

string **ncurses_termname** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_termname() returns terminals shortname. The shortname is truncated to 14 characters. On error **ncurses_termname()** returns NULL.

See also: ncurses_longname()

ncurses_longname (PHP 4 CVS only)

Returns terminals description

string **ncurses_longname** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_longname() returns a verbose description of the terminal. The descritpion is truncated to 128 characters. On Error **ncurses_longname()** returns NULL.

See also: ncurses_termname()

ncurses_mousemask (PHP 4 CVS only)

Sets mouse options

int **ncurses_mousemask** (int newmask, int oldmask)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Function **ncurses_mousemask()** will set mouse events to be reported. By default no mouse events will be reported. The function **ncurses_mousemask()** will return a mask to indicated which of the in parameter `newmask` specified mouse events can be reported. On complete failure, it returns 0. In parameter `oldmask`, which is passed by reference **ncurses_mousemask()** returns the previous value of mouse event mask. Mouse events are represented bei NCURSES_KEY_MOUSE in the

ncurses_wgetch() input stream. To read the event data and pop the event of of queue, call ncurses_getmouse().

As a side effect, setting a zero mousemask in `newmask` turns off the mouse pointer. Setting a non zero value turns mouse pointer on.

mouse mask options can be set with the following predefined constants:

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED
- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

See also: ncurses_getmouse(), ncurses_ungetmouse() **ncurese_getch()**

Beispiel 1. ncurses_mousemask() example

```
$newmask = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mask = ncurses_mousemask($newmask, &$oldmask);
if ($mask & $newmask){
```

```
  printf ("All specified mouse options will be supported\n");
}
```

ncurses_getmouse (PHP 4 CVS only)

Reads mouse event

bool **ncurses_getmouse** (array mevent)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_getmouse() reads mouse event out of queue. Function **ncurses_getmouse()** will return ;`FALSE` if a mouse event is actually visible in the given window, otherwise it will return `TRUE`. Event options will be delivered in parameter *`mevent`*, which has to be an array, passed by reference (see example below). On success an associative array with following keys will be delivered:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

Beispiel 1. ncurses_getmouse() example

```
switch (ncurses_getch){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; // Save mouse position
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    ....
```

```
}
```

See also: ncurses_ungetmouse()

ncurses_ungetmouse (PHP 4 CVS only)

Pushes mouse event to queue

bool **ncurses_ungetmouse** (array mevent)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

ncurses_getmouse() pushes a KEY_MOUSE event onto the unput queue and associates with this event the given state sata and screen-relative character cell coordinates, specified in *mevent*. Event options will be specified in associative array *mevent*:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

ncurses_ungetmouse() returns FALSE on success, otherwise TRUE.

See also: ncurses_getmouse()

LXVI. Lotus Notes functions

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

notes_create_db (PHP 4 >= 4.0.5)

Create a Lotus Notes database

bool **notes_create_db** (string database_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_drop_db (PHP 4 >= 4.0.5)

Drop a Lotus Notes database

bool **notes_drop_db** (string database_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_version (PHP 4 >= 4.0.5)

Get the version Lotus Notes

string **notes_version** (string database_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_create_note (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_create_note** (string database_name, string form_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_mark_read (PHP 4 >= 4.0.5)

Mark a note_id as read for the User user_name

string **notes_mark_read** (string database_name, string user_name, string note_id)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_mark_unread (PHP 4 >= 4.0.5)

Mark a note_id as unread for the User user_name

string **notes_mark_unread** (string database_name, string user_name, string note_id)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_unread (PHP 4 >= 4.0.5)

Returns the unread note id's for the current User user_name

string **notes_unread** (string database_name, string user_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_header_info (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

object **notes_header_info** (string server, string mailbox, int msg_number)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_body (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

array **notes_body** (string server, string mailbox, int msg_number)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_find_note (PHP 4 >= 4.0.5)

Returns a note id found in database_name. Specify the name of the note. Leaving type bla

bool **notes_find_note** (string database_name, string name [, string type])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_nav_create (PHP 4 >= 4.0.5)

Create a navigator name, in database_name

bool **notes_nav_create** (string database_name, string name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_search (PHP 4 >= 4.0.5)

Find notes that match keywords in database_name

string **notes_search** (string database_name, string keywords)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_copy_db (PHP 4 >= 4.0.5)

title]) Create a note using form form_name

string **notes_copy_db** (string from_database_name, string to_database_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

notes_list_msgs (PHP 4 >= 4.0.5)

Returns the notes from a selected database_name

bool **notes_list_msgs** (string db)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LXVII. ODBC Funktionen

odbc_autocommit (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ändert das Autocommit-Verhalten

int **odbc_autocommit** (int connection_id [, int OnOff])

Ohne den Parameter *OnOff* gibt **odbc_autocommit()** den aktuellen autocommit-Status für die Datenbankverbindung *connection_id* zurück. Wenn autocommit aktiviert ist, wird `TRUE` zurückgeliefert, `FALSE` wenn es deaktiviert ist oder ein Fehler auftritt.

In der Standardeinstellung ist autocommit für eine Datenbankverbindung aktiviert. Das Deaktivieren von autocommit enspricht dem Durchführen einer Transaktion.

Siehe auch odbc_commit() und odbc_rollback().

odbc_binmode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Die Behandlung von Binärdaten

int **odbc_binmode** (int result_id, int mode)

(Betrifft die ODBC SQL Typen: BINARY, VARBINARY und LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: gibt Binärdaten direkt zum Output durch
- ODBC_BINMODE_RETURN: liefert die Binärdaten unverändert zurück
- ODBC_BINMODE_CONVERT: konvertiert die Binärdaten zu Zeichen und liefere sie zurück

Im `ODBC_BINMODE_CONVERT` werden die Binärdaten zu Character-C-data gewandelt. Jedes Byte (8 Bits) der ursprünglichen Daten wird zu zwei ASCII-Zeichen, die der hexadezimalen Darstellung des Wertes entsprechen. Der Binärwert 00000001 wird zum Beispiel zu `"01"` gewandelt, der Binärwert 11111111 zu `"FF"`.

Tabelle 1. Die Benutzung von LONGVARBINARY

binmode	longreadlen	Resultat
ODBC_BINMODE_PASSTHRU	0	Direkte Ausgabe
ODBC_BINMODE_RETURN	0	Direkte Ausgabe
ODBC_BINMODE_CONVERT	0	Direkte Ausgabe
ODBC_BINMODE_PASSTHRU	0	Direkte Ausgabe
ODBC_BINMODE_PASSTHRU	>0	Direkte Ausgabe
ODBC_BINMODE_RETURN	>0	Direkte Ausgabe
ODBC_BINMODE_CONVERT	>0	Rückgabe als CHAR-Werte

Wenn odbc_fetch_into() genutzt wird, wird bei direkter Ausgabe ein leerer String als Spaltenwert zurückgeliefert.

Wenn `result_id` gleich `0` ist, wird die Einstellung als Defaultwert für die nächsten Transaktionen gesetzt.

> **Anmerkung:** Der Defahltwert für longreadlen ist `4096`, für **odbc_binmode()** `ODBC_BINMODE_RETURN`, also die unveränderte Rückgabe. Das Verhalten von `LONGVARBINARY` wird ebenfalls durch die Funktion odbc_longreadlen() bestimmt.

odbc_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Beendet eine ODBC-Verbindung

void **odbc_close** (int connection_id)

odbc_close() beendete die Verbindung zum Datenbankserver, die durch das Verbindungskennzeichen `connection_id` gekennzeichnet wird.

> **Anmerkung:** Diese Funktion scheitert, wenn auf dieser Datenbankverbindung noch Transaktionen ausstehen. In diesem Fall bleibt die Verbindung bestehen.

odbc_close_all (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Beendet alle ODBC-Verbindungen

void **odbc_close_all** (void)

odbc_close_all() beendet alle bestehenden Verbindungen zu allen Datenbankservern.

> **Anmerkung:** Wie odbc_close() scheitert auch dieser Befehl, wenn auf einer Datenbankverbindung noch Transaktionen ausstehen oder aktiv sind. In diesem Fall bleibt die betroffene Verbindung auch nach dem **odbc_close_all()** bestehen.

odbc_commit (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Führt eine ODBC-Transaktion aus

int **odbc_commit** (int connection_id)

Gibt bei Erfolg `TRUE` zurück, bei Mißerfolg `FALSE`. Alle ausstehenden Transaktionen auf der Datenbankverbindung *`connection_id`* werden ausgeführt.

odbc_connect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Baut die Verbindung zu einer ODBC-Datenquelle auf

int **odbc_connect** (string dsn, string user, string password [, int cursor_type])

Liefert eine ODBC-Verbindungskennung `connection_id` oder 0 (`FALSE`) bei Fehlfunktion zurück.

Die Verbindungskennung `connection_id` wird von den anderen ODBC-Funktionen benötigt. Es können mehrere ODBC-Verbindungen nebeneinander bestehen. Der optionale vierte Parameter `cursor_type` setzt die Art des Cursors, der für diese Verbindung genutzt werden soll. Normalerweise wird er nicht benötigt, kann aber bei manchen ODBC-Treiberproblemen nützlich sein.

Es können etwa Probleme bei der Ausführung von komplexen Prozeduren auftreten. Die Fehlermeldung lautet dann zum Beispiel "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Dieser Fehler kann durch die Verwendung von `SQL_CUR_USE_ODBC` als `cursor_type` kann hier helfen. Einige ODBC-Treiber unterstützen den optionalen Parameter `row_number` in der Funktion odbc_fetch_row() nicht. Auch in diesem Fall kann `SQL_CUR_USE_ODBC` helfen.

Folgende Konstanten sind als Cursortyp definiert:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

Für persistente Verbindungen schauen Sie sich bitte odbc_pconnect() an.

odbc_cursor (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Findet den Cursornamen heraus

string **odbc_cursor** (int result_id)

odbc_cursor() liefert einen Cursornamen für die gegebene Ergebniskennung *`result_id`* zurück.

odbc_do (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ein Synonym für odbc_exec()

int **odbc_do** (int conn_id, string query)

odbc_do() führt eine Anfrage auf der gegebenen Verbindung durch.

odbc_exec (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Bereitet einen SQL-Befehl auf und führt ihn aus

int **odbc_exec** (int connection_id, string query_string)

Liefert bei einem Fehler `FALSE` zurück, sonst eine ODBC-Ergebniskennung `result_id`.

odbc_exec() sendet einen SQL-Befehl zu dem Datenbankserver, der durch die Verbindungskennung `connection_id` bezeichnet wird. Dieser Parameter muß durch odbc_connect() oder odbc_pconnect() erzeugt worden sein.

Siehe auch: odbc_prepare() und odbc_execute() für die gleichzeitige Ausführung von mehreren SQL-Befehlen.

odbc_execute (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Führt ein vorbereiteten SQL-Befehl aus

int **odbc_execute** (int result_id [, array parameters_array])

Führt ein oder mehrere durch odbc_prepare() vorbereitete SQL-Befehle aus. Nach erfolgreicher Ausführung wird `TRUE` zurückgeliefert, sonst `FALSE`. Das Array *`parameters_array`* braucht nur angegeben zu werden, wenn im Befehl tatsächlich Parameter vorhanden sind.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Eine Ergebniszeile in ein Array stellen

int **odbc_fetch_into** (int result_id [, int rownumber, array result_array])

Liefert die Anzahl der Ergebnisspalten zurück, bei einem Fehler `FALSE`. Das Zielarray *`result_array`* muß immer übergeben werden, kann aber einen beliebigen Typ haben - es wird ohnehin zu einem Array gemacht. Das Array enthält die Ergebnisspalten beginnend mit `0`.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert eine Datenzeile zurück

int **odbc_fetch_row** (int result_id [, int row_number])

Wenn **odbc_fetch_row()** erfolgreich durchgeführt wurde (also eine oder mehrere Datenzeilen gefunden wurden), dann wird `TRUE` zurückgeliefert. Wenn keine (weitere) Zeile vorhanden ist, wird `FALSE` zurückgegeben.

odbc_fetch_row() stellt eine Zeile der von odbc_do()/odbc_exec() zurückgegeben Daten bereit. Nachdem **odbc_fetch_row()** aufgerufen wurde, können die Felder dieser Zeile mit odbc_result() genutzt werden.

Wenn *row_number* nicht definiert ist, versucht **odbc_fetch_row()** die nächste Zeile aus dem Abfrageergebnis `result_id` zu holen. **odbc_fetch_row()** kann abwechselnd ohne und mit dem Parameter *row_number* genutzt werden.

Um durch ein Abfrageergebnis mehrfach zu durchlaufen, kann man **odbc_fetch_row()** mit *row_number* gleich 1 aufrufen, um dann wie gewohnt mit **odbc_fetch_row()** ohne Zeilenparameter fortzufahren. Wenn der ODBC-Treiber die Herausgabe von bestimmten Zeilen nicht unterstützt, wird der Parameter ignoriert.

odbc_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert die Spaltenbezeichnung

string **odbc_field_name** (int result_id, int field_number)

odbc_field_name() gibt den Namen der Spalte mit der Nummer `field_number` für des Abfrageergebnisses `result_id` zurück. Die Nummerierung der Spalten beginnt mit 1. Bei einem Fehler wird `FALSE` zurückgegeben.

odbc_field_type (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert den Datentyp eines Feldes

string **odbc_field_type** (int result_id, int field_number)

odbc_field_type() liefert den SQL-Datentyp eines Feldes zurück, das in dem Abfrageergebnis *result_id* an der Stelle *field_number* zu finden ist.

odbc_field_len (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Bestimmt die Länge eines Feldes

int **odbc_field_len** (int result_id, int field_number)

odbc_field_len() gibt die Länge eines Feldes an. Das Feld wird durch die Feldnummer *field_number* im Abfrageergebnis *result_id* bestimmt. Die Feldnummerierung beginnt mit 1.

odbc_free_result (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt den durch ein Abfrageergebnis belegten Speicher wieder frei

int **odbc_free_result** (int result_id)

Gibt immer TRUE zurück.

odbc_free_result() braucht nur aufgerufen zu werden, wenn Sie einen zu hohen Speicherverbrauch durch das aktuell laufende Skript vermeiden wollen. Das gesamte Abfrageergebnis wird automatisch nach Beendigung des Skriptes gelöscht. Aber wenn Sie sicher sind, dass Sie dieses Ergebnis nicht länger benötigen, dann können Sie bei Speicherproblemen die Funktion **odbc_free_result()** benutzen.

Anmerkung: Wenn autocommit deaktiviert ist (siehe odbc_autocommit()) und Sie **odbc_free_result()** bevor ein Commit gemacht wird, werden alle austehenden Transaktionen zurückgenommen.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Steuert die Nutzung von LONG-Spalten

int **odbc_longreadlen** (int result_id, int length)

(Betrifft folgende ODBC-SQL-Typen: LONG, LONGVARBINARY) Die Anzahl der an PHP zurückgelieferten Bytes wird von durch den Parameter *length* bestimmt. Wenn er auf 0 gesetzt ist, wird der Inhalt von LONG-Spalten direkt an den Klienten weitergegeben.

Anmerkung: Die Behandlung von LONGVARBINARY-Spalten wird auch von odbc_binmode() beeinflußt.

odbc_num_fields (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert die Anzahl der Ergebnisspalten

int **odbc_num_fields** (int result_id)

odbc_num_fields() liefert die Anzahl der Spalten, die in einem ODBC-Ergebnis vorhanden sind, zurück. Bei einem Fehler ergibt **odbc_num_fields()** -1.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Öffnet eine persistente Datenbankverbindung

int **odbc_pconnect** (string dsn, string user, string password [, int cursor_type])

Gibt eine ODBC-Verbindungskennung zurück, bei einem Fehler 0 (FALSE). Diese Funktion ist odbc_connect() sehr ähnlich. Der einzige Unterschied besteht darin, dass die Verbindung nach dem Ende des Skriptes erhalten bleibt. Folgende Verbindungen mit gleicher Datenquelle, Nutzer und Passwort-Kombination über odbc_connect() oder **odbc_pconnect()** können dann die bestehende Verbindung weiterbenutzen.

> **Anmerkung:** Persistente Verbindungen können bei einem CGI-PHP nicht benutzt werden. In diesem Fall entspricht **odbc_pconnect()** odbc_connect().

Für weitere Informationen über den optionalen Parameter *cursor_type* schauen Sie bitte bei odbc_connect(). Weitere Informationen über persistente Datenbankverbindungen finden Sie in der PHP-FAQ

odbc_prepare (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Stellt einen SQL-Befehl zur Ausführung bereit

int **odbc_prepare** (int connection_id, string query_string)

Liefert bei einem Fehler FALSE zurück.

Gibt eine ODBC-Ergebniskennung zurück, wenn der SQL-Befehl erfolgreich bereitgestellt wurde. Die Ergebniskennung kann später zur Ausführung des Befehles mit odbc_execute() genutzt werden.

odbc_num_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ergibt die Zeilenzahl des Abfrageergebnisses

int **odbc_num_rows** (int result_id)

odbc_num_rows() liefert die Anzahl Zeilen eines ODBC-Abfrageergebnisses. Tritt ein Fehler auf, ist sie -1. Bei INSERT, UPDATE und DELETE-Befehlen wird die Zahl der betroffenen Zeilen zurückgegeben. Bei einer SELECT-Abfrage kann dies die Anzahl der vorhanden Zeilen sein.

> **Anmerkung:** Bei vielen Treibern wird -1 als Anzahl möglicher Zeilen nach einem SELECT-Befehl zurückgeliefert.

odbc_result (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Erlaubt den Zugriff auf die Ergebnisdaten

string **odbc_result** (int result_id, mixed field)

Liefert den Inhalt eines bestimmtes Feldes zurück.

Der Feldparameter *field* kann entweder der Integerwert der Spaltennummer oder ein String mit dem Spaltennamen sein. Als Beispiel:

```
$produkt_3 = odbc_result ($Query_ID, 3 );
$produkt_wert = odbc_result ($Query_ID, "wert");
```

Der erste Aufruf von **odbc_result()** liefert den Wert des drittes Feldes im aktuellen Datensatz im Abfrageergebnis zurück. Der zweite Funktionsaufruf gibt den Wert der Spalte "wert" im aktuellen Datensatz zurück. Wenn die Spaltennummer kleiner als 1 oder größer als die Spaltenzahl ist, tritt ein Fehler auf. Gleiches gilt für einen Spaltennamen, der im Abfrageergebnis nicht vorkommt.

Die Spaltennummern beginngen mit 1. Für die Nutzung von Binärdaten und LONG-Spalten schauen Sie bitte bei odbc_binmode() und odbc_longreadlen() nach.

odbc_result_all (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gibt das aktuelle Abfrageergebnis als HTML-Tabelle aus

int **odbc_result_all** (int result_id [, string format])

Liefert die Anzahl Reihen im Abfrageergebnis `result_id` zurück, wenn ein Fehler auftritt `FALSE`.

odbc_result_all() gibt alle Reihen eines SQL-Abfrageergebnisses `result_id` aus einem odbc_exec()-Aufruf als HTML-Tabelle aus. Mit dem optionalen String *format* kann ein allgemeines Tabellenformat angegeben werden.

odbc_rollback (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Hebt eine Transaktion wieder auf

int **odbc_rollback** (int connection_id)

Hebt alle ausstehenden Transaktionen auf der Verbindung *connection_id* wieder auf. Bei Erfolg wird `TRUE` zurückgegeben, sonst `FALSE`.

odbc_setoption (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Verändert die ODBC-Einstellungen

int **odbc_setoption** (int id, int function, int option, int param)

Diese Funktion erlaubt das Einstellen der ODBC-Optionen für eine einzelne Verbindung. Sie soll beim Auffinden und Beheben von Problemen mit ODBC-Treibern helfen. Es empfiehlt sich, diese Funktionen nur zu nutzen, wenn man einige Erfahrung mit ODBC-Programmierung gesammelt hat und die Wirkungen der einzelnen Parameter kennt. Um mit Ihnen zu arbeiten, benötigen Sie eine

Referenz der aktuellen Treiberversion des eingesetzten ODBC-Treibers, denn die Optionen sind von Treiber zu Treiber und von Version zu Version unterschiedlich.

Weil die Effekte je nach ODBC-Treiber unterschiedlich sind, sollte Skripte mit diesen Funktionen nicht veröffentlich werden. Einige Optionen stehen nicht zur Verfügung, weil sie vor dem Aufbau einer ODBC-Verbindung eingestellt sein müssen. Wie auch immer, in einigen Fällen hilft diese Funktion zu vermeiden, dass man von seinem Vorgesetzten gezwungen wird, ein kommerzielles Programm einzusetzen - und darauf kommt es doch an ;-) .

`Id` ist die Verbindungs oder Ergebniskennung, deren Optionen geändert werden sollen. Bei SQLSetConnectOption() ist es eine Verbindungskennung, bei SQLSetStmtOption() eine Ergebniskennung.

`function` ist die ODBC-Funktion, die genutzt werden soll. Die Werte sind 1 für SQLSetConnectOption() und 2 für SQLSetStntOption().

Der Parameter *`option`* ist die Option, die gesetzt werden soll.

Der Parameter *`param`* ist der Wert für die gegebene *`option`*.

Beispiel 1. odbc.setoption Beispiele

```
// 1. Option 102 von SQLSetConnectOption() ist SQL_AUTOCOMMIT.
//    Wert 1 für SQL_AUTOCOMMIT entspricht SQL_AUTOCOMMIT_ON.
//    Dieses Beispiel hat denselben Effekt wie
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Option 0 von SQLSetStmtOption() ist SQL_QUERY_TIMEOUT.
//    Dieses Beispiel setzt den Timeout auf 30 Sekunden.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

LXVIII. Oracle 8 Funktionen

Diese Funktionen gestatten den Zugriff auf Oracle7 und Oracle8 Datenbanken. Es wird hierbei das Oracle8 Call-Interface (OCI8) verwendet, welches installierte Oracle8 Client Bibliotheken (Libraries) benötigt.

Diese Schnittstelle ist flexibler als die Standard Oracle Funktionen. Es unterstützt Binding von localen und globalen PHP Variablen an Oracle Platzhalter und hat volle LOB,FILE und ROWID Unterstützung und gestattet dem Benutzer die Verwendung von Variablendefines.

Vor der Benutzung dieser Erweiterung sollten Sie sicherstellen die folgenden Umgebungsvariablen für den Webserveruser VOR dem Start des Webservers gesetzt zu haben.

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

Nachdem Sie die Umgebungsvariablen festgelegt haben, stellen Sie sicher, das die Dateien im ORACLE_HOME/network/admin für den Webserver lesbar sind.

Wenn der Webserver abstürzt nach dem Start, oder nicht startet: Überprüfen Sie, ob der Apache gegen -lpthread gelink ist:

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Wenn libpthread hier nicht gelistet ist, dann muss der Webserver neu übersetzt werden:

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Beispiel 1. OCI Hinweise

```
<?php
// by sergo@bacup.ru

// Die Option OCI_DEFAULT führt die Anweisung in einer Transaktion aus
OCIExecute($stmt, OCI_DEFAULT);

// for retrieve data use (after fetch):

$result = OCIResult($stmt, $n);
if (is_object ($result)) $result = $result->load();

// For INSERT or UPDATE statement use:

$sql = "insert into table (field1, field2) values (field1 = 'value',
 field2 = empty_clob()) returning field2 into :field2";
OCIParse($conn, $sql);
$clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName ($stmt, ":field2", &$clob, -1, OCI_B_CLOB);
OCIExecute($stmt, OCI_DEFAULT);
$clob->save ("some text");
OCICommit($conn);

?>
```

Man kann auf Stored Procedures auf einfache Weise zugreifen.

Beispiel 2. Stored Procedures benutzen

```
<?php
// by webmaster@remoterealty.com
$sth = OCIParse ( $dbh, "begin sp_newaddress( :address_id, '$firstname',
 '$lastname', '$company', '$address1', '$address2', '$city', '$state',
 '$postalcode', '$country', :error_code );end;" );

// This calls stored procedure sp_newaddress, with :address_id being an
// in/out variable and :error_code being an out variable.
// Then you do the binding:

   OCIBindByName ( $sth, ":address_id", $addr_id, 10 );
   OCIBindByName ( $sth, ":error_code", $errorcode, 10 );
   OCIExecute ( $sth );

?>
```

OCIDefineByName (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Definiert eine PHP Variable als Platzhalter für die Werte eines SELECTs

int **OCIDefineByName** (int stmt, string Spaltenname, mixed & Variable [, int Typ])

OCIDefineByName() legt eine PHP Variable als Ziel für den Inhalt einer Spalte nach dem Fetchen fest. Es ist zu beachten, dass Oracle alle Spalten GROSSBUCHSTABIG ausliefert. Daher erwartet **OCIDefineByName()** den Parameter *Spaltenname* grossbuchstabig. Wird ein *Spaltenname* benutzt, der nicht im SELECT vorhanden ist, wird keine Fehlermeldung ausgegeben.

Soll ein abstrakter Datentyp (LOB/ROWID/BFILE) definiert werden, so muss er zuerst angefordert werden mit der **OCINewDescriptor()** Funktion. In diesem Zusammenhang ist auch die **OCIBindByName()** zu beachten.

Beispiel 1. OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@thieso.net (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* the define MUST be done BEFORE ociexecute! */

OCIDefineByName($stmt,"EMPNO",&$empno);
OCIDefineByName($stmt,"ENAME",&$ename);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIBindByName (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Bindet eine PHP-Variable an einen Oracle Platzhalter

int **OCIBindByName** (int stmt, string Platzhaltername, mixed & Variable, int Länge [, int Typ])

OCIBindByName() bindet die PHP Variable *variable* an den Oracle Platzhalter *ph_name*. Es wird genutzt um den benötigten Speicherplatz fuer die Ein/Ausgabe zu Laufzeit festzulegen. *length* gibt dabei die maximale Grösse des Speicherplatzes an. Wenn *length* auf -1 gesetzt wird, nimmt **OCIBindByName()** die Grösse des Inhalts von *variable* als maximale Grösse.

Wenn ein abstrakter Datentyp wie (LOB/ROWID/BFILE) gebunden werden soll, muss zuerst **OCINewDescriptor()** verwendet werden. *length* gilt nicht für abstrakte Datentypen und sollte

denn auf -1 gesetzt sein. *type* zeigt Oracle, welchen Typ wir binden wollen. Mögliche Typen sind: OCI_B_FILE (Binärdatei), OCI_B_CFILE (Textdatei), OCI_B_CLOB (Text-LOB), OCI_B_BLOB (Binär-LOB) and OCI_B_ROWID (ROWID).

Beispiel 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@thieso.net (980221)

  trägt 3 Datensätze in emp ein und benutzt ROWID für das UPDATE nach dem
  INSERT
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
   "values (:empno,:ename) ".
   "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);

OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
OCIExecute($stmt);
OCIExecute($update);
}

$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

OCILogon (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Baut eine OCI-Verbindung auf

int **OCILogon** (string Benutzername, string Passwort [, string Datenbankname])

OCILogon() gibt bei erfolgreicher Verbindung einen Verbindungsindex zurück. Bei einem Fehler `FALSE`. Der optionale dritte Parameter kann auf die lokale Oracleinstanz gesetzt werden oder auf die, zu welcher sich verbunden werden soll. Diese sollte in der tnsnames.ora aufgeführt sein. Wird der dritte Parameter nicht gesetzt nimmt PHP automatisch den Wert aus der Umgebungsvariable ORACLE_SID oder TWO_TASK.

Verbindungen werden gemeinsam wärend eines Skriptes genutzt, wenn **OCILogon()** benutzt wird. Das heisst, dass commits und rollbacks sich auf alle offenen Transaktionen im Skript auswirken, ausser wenn explizit mehrere Verbindungen aufgebaut werden.

Dies Beispiel zeigt, wie die Verbindungen gemeinsam genutzt werden.

Beispiel 1. OCILogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo values('$conn' || ' ' || to_char(sysdat
MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." commited\n\n";
}
```

```
function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);   // trägt eine Zeile ein und benutzt dabei c1
insert_data($c2);   // trägt eine Zeile ein und benutzt dabei c2

select_data($c1);   // die Ergebnisse beider Eintragungen wird zurückgegeben
select_data($c2);

rollback($c1);      // Rollback benutzt c1

select_data($c1);   // ein Rollback auf beide Eintragungen
select_data($c2);

insert_data($c2);   // trägt eine Zeile ein und benutzt dabei c2
commit($c2);        // der commit für c2

select_data($c1);   // das Ergebnis der Eintragung über c2 wird zurückgegeben

delete_data($c1);   // löschen aller Zeilen über c1
select_data($c1);   // keine Zeile zurückgegeben
select_data($c2);   // keine Zeile zurückgegeben
commit($c1);        // commit für c1

select_data($c1);   // keine Zeile zurückgegeben
select_data($c2);   // keine Zeile zurückgegeben


drop_table($c1);
print "</PRE></HTML>";
?>
```

Bitte beachte auch **OCIPLogon()** und **OCINLogon()**.

OCIPLogon (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Logt sich persistent in die Oracle Datenbank ein und erzeugt eine neue Oracle Session.

int **OCIPLogon** (string username, string password [, string Datenbank])

OCIPLogon() gibt bei erfolgreicher Öffnung einer persistenten Verbindung einen Verbindungsindex zurück. Bei einem Fehler `FALSE`. Der optionale dritte Parameter kann auf die lokale Oracleinstanz gesetzt werden oder auf die, zu welcher sich verbunden werden soll. Diese sollte in der tnsnames.ora aufgeführt sein. Wird der dritte Parameter nicht gesetzt nimmt PHP automatisch den Wert aus der Umgebungsvariable ORACLE_SID oder TWO_TASK.

Bitte ebenfalls beachten **OCILogon()** und **OCINLogon()**.

OCINLogon (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Logt sich in die Oracle Datenbank ein und erzeugt dabei eine neue Oracle Session.

int **OCINLogon** (string username, string password [, string db])

OCINogon() gibt bei erfolgreicher Verbindung einen Verbindungsindex zurück. Bei einem Fehler `FALSE`. Der optionale dritte Parameter kann auf die lokale Oracleinstanz gesetzt werden oder auf die, zu welcher sich verbunden werden soll. Diese sollte in der tnsnames.ora aufgeführt sein. Wird der dritte Parameter nicht gesetzt nimmt PHP automatisch den Wert aus der Umgebungsvariable ORACLE_SID oder TWO_TASK.

OCINLogon() erzeugt eine neue Verbindung mit einer neuen Oracle Session, dies ist zum Beispiel erforderlich, wenn man mit voneinander getrennten Transaktionen arbeitet. Normalerweise wird die Oracle Session gemeinsam genutzt, wenn man **OCILogon()** oder auf Webserverebene **OCIPLogon()** . Wenn man allerdings unterschiedliche Verbindungen mit **OCINLogon()** öffnet, beeinflussen sich die Commits und Rollbacks der einzelnen Datenbankverbindungen nicht.

Dieses Beispiel zeigt, das es sich um unterschiedliche Verbindungen handelt:

Beispiel 1. OCINLogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo values('$conn' || ' ' || to_char(sysdat
MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
```

```
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." commited\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);


drop_table($c1);
print "</PRE></HTML>";
?>
```

Bitte beachten **OCILogon()** und **OCIPLogon()**.

OCILogOff (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Schliesst eine Datenbankverbindung zu Oracle

int **OCILogOff** (int Verbindungskennung)

OCILogOff() schliesst die Verbindung zu Oracle.

OCIExecute (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

führt ein Statement aus.

int **OCIExecute** (int statement [, int mode])

OCIExecute() führt ein vorbereitetes Statement aus (beachte **OCIParse()**). Der optionale Parameter *mode* ermöglicht die Wahl des Ausführungsmodus. Voreingestellt ist OCI_COMMIT_ON_SUCCESS, wenn das Statement aber nicht gleich endgültig gemacht werden soll, ist OCI_DEFAULT zu verwenden.

OCICommit (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Committet eine ausstehende Transaktion.

int **OCICommit** (int Verbindungskennung)

OCICommit() schreibt alle Datenbankänderungen seit dem letzten implizit oder explizitem Commit der *Verbindungskennung* fest.

OCIRollback (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Macht ausstehende Transaktionen rückgängig

int **OCIRollback** (int Verbindungskennung)

OCIRollback() macht alle Änderungen seid dem letzten Commit rückgängig und gibt auch Sperren auf den Tabellen frei. Mit der *Verbindungskennung* legt man die Datenbankverbindung fest, für welche diese Aktion gelten soll.

OCINewDescriptor (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Initialize a new empty descriptor LOB/FILE (LOB is default)

string **OCINewDescriptor** (int connection [, int type])

OCINewDescriptor() Allocates storage to hold descriptors or LOB locators. Valid values for the valid *type* are OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. For LOB desriptors, the methods load, save, and savefile are associated with the descriptor, for BFILE only the load method exists. See the second example usage hints.

Beispiel 1. OCINewDescriptor

```
<?php
    /* This script is designed to be called from a HTML form.
     * It expects $user, $password, $table, $where, and $commitsize
     * to be passed in from the form.  The script then deletes
     * the selected rows using the ROWID and commits after each
     * set of $commitsize rows. (Use with care, there is no rollback)
     */
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn,"select rowid from $table $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
       $nrows = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $table where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           OCICommit($conn);
       }
    }
    $nrows = OCIRowCount($stmt);
    print "$nrows deleted...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>

<?php
    /* This script demonstrates file upload to LOB columns
     * The formfield used for this example looks like this
     * <form action="upload.php3" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
  } else {
     // $lob_upload contains the temporary filename of the uploaded file
     $conn = OCILogon($user, $password);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
     $stmt = OCIParse($conn,"insert into $table (id, the_blob) values(my_seq.NEXTVAL, EMP
turning the_blob into :the_blob");
```

```
      OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
      OCIExecute($stmt);
      if($lob->savefile($lob_upload)){
         OCICommit($conn);
         echo "Blob successfully uploaded\n";
      }else{
         echo "Couldn't upload Blob\n";
      }
      OCIFreeDescriptor($lob);
      OCIFreeStatement($stmt);
      OCILogoff($conn);
   }
?>
```

OCIRowCount (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt die Anzahl der betroffenen Zeilen zurück

int **OCIRowCount** (int statement)

OCIRowCounts() gibt die Anzahl der betroffenen Zeilen zum Beispiel einer UPDATE Aktion zurück. Sie gibt nicht die Anzahl der zurückgelieferten Zeilen von einem SELECT Statements zurück!

Beispiel 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table emp2 as select * from emp");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows inserted.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from emp2");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows deleted.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table emp2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

OCINumCols (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Gibt die Anzahl der Spalten in einem Statement zurück

int **OCINumCols** (int stmt)

OCINumCols() gibt die Anzahl der Spalten in einem Statement zurück.

Beispiel 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = OCIColumnName($stmt,$i);
            $column_value = OCIResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIResult (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Gibt einen Spaltenwert für das gefetchte Statement zurück

mixed **OCIResult** (int statement, mixed Spaltenname)

OCIResult() gibt den Inhalt der Spalte mit dem *`Spaltenname`* in der aktuellen Zeile zurück (dazu wichtig **OCIFetch()**).**OCIResult()** gibt alles als Zeichenkette zurück, ausser es sind abstrakte Datentypen (ROWIDs, LOBs and FILEs).

OCIFetch (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Fetcht die nächste Ergebniszeile in den Ergebnispuffer

int **OCIFetch** (int statement)

OCIFetch() fetcht die nächste Ergebniszeile eines SELECTs in den internen Ergebnispuffer.

OCIFetchInto (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Fetcht die nächste Zeile eines Ergebnisses in ein Array

int **OCIFetchInto** (int stmt, array & result [, int mode])

OCIFetchInto() fetcht die nächste Zeile eines SELECT Ergebnisses in das *result* Array. **OCIFetchInto()** überschreibt dabei den Inhalt in *result*. Voreingestellt beinhaltet dann *result* ein bei 0 beginnendes indiziertes Array mit allen Spalten, welche nicht `NULL` sind.

Der *mode* Parameter erlaubt die Art der Form des *result* Arrays zu ändern. Dabei können verschiedene Parameter benutzt werden, es reicht, diese einfach zu addieren (z.B. OCI_ASSOC+OCI_RETURN_NULLS). Verfügbare Parameter sind:

`OCI_ASSOC` Gibt ein assoziatives Array zurück.
`OCI_NUM` Gibt ein indiziertes Array zurück, beginnend bei 0. (voreingestellt)
`OCI_RETURN_NULLS` Gibt auch leere Spalten zurück.
`OCI_RETURN_LOBS` Gibt den Wert eines LOBs zurück, anstatt nur des Descriptors.

OCIFetchStatement (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetcht die Ergebnisse in ein Array

int **OCIFetchStatement** (int stmt, array & variable)

OCIFetchStatement() fetcht alle zurückgelieferten Reihen eines SELECT in ein Array. Dabei liefert **OCIFetchStatement()** die Anzahl der zurückgelieferten Zeilen zurück.

Beispiel 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
```

```
            print "<TD>$data[$i]</TD>\n";
         }
         print "</TR>\n";
      }
      print "</TABLE>\n";
   } else {
      echo "No data found<BR>\n";
   }
   print "$nrows Records Selected<BR>\n";

   OCIFreeStatement($stmt);
   OCILogoff($conn);

   ?>
```

OCIColumnIsNULL (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Testet eine Spalte ob deren Inhalt `NULL` ist

int **OCIColumnIsNULL** (int stmt, mixed column)

OCIColumnIsNULL() gibt `TRUE` zurück, wenn die Spalte *`col`* vom Statement *`stmt`* `NULL` ist. Für den Parameter *`column`* kann wahlweise die Splatennummer (beginnend bei 1) oder der Spaltenname genommen werden.

OCIColumnSize (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Gibt die Grösse einer Spalte zurück

int **OCIColumnSize** (int stmt, mixed column)

OCIColumnSize() gibt die grösse einer Oracle Spalte zurück. Für den Parameter *`column`* kann wahlweise die Splatennummer (beginnend bei 1) oder der Spaltenname genommen werden.

Beispiel 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
```

```
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Beachte bitte auch **OCINumCols()**, **OCIColumnName()**, und **OCIColumnSize()**.

OCIServerVersion (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Return a string containing server version information.

string **OCIServerVersion** (int conn)

Beispiel 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
   print "Server Version: " . OCIServerVersion($conn);
   OCILogOff($conn);
?>
```

OCIStatementType (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Gibt den Typ eine Statements zurück

string **OCIStatementType** (int stmt)

OCIStatementType() kann folgende Werte liefern:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"

8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Beispiel 1. Beispiel:

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
        die "You are not allowed to delete from this table<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

OCINewCursor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

return a new cursor (Statement-Handle) - use this to bind ref-cursors!

int **OCINewCursor** (int conn)

OCINewCursor() allocates a new statement handle on the specified connection.

Beispiel 1. Using a REF CURSOR from a stored procedure

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

Beispiel 2. Using a REF CURSOR in a select statement

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
print "<TH># EMPLOYEES</TH>";
print "</TR>";

while (OCIFetchInto($stmt,&$data,OCI_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data[ "EMPCNT" ]);
    while (OCIFetchInto($data[ "EMPCNT" ],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIFreeStatement (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Gibt alle mit dem Statement verbundenen Ressourcen frei.

int **OCIFreeStatement** (int stmt)

OCIFreeStatement() gibt `TRUE` zurück, wenn es erfolgreich war, `FALSE`, wenn nicht.

OCIFreeCursor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gibt alle mit dem cursor verbundenen Ressourcen frei.

int **OCIFreeCursor** (int stmt)

OCIFreeCursor() gibt TRUE zurück, wenn die Aktion erfolgreich war, FALSE, wenn nicht.

OCIColumnName (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Liefert den Namen der Spalte zurück.

string **OCIColumnName** (int stmt, int col)

OCIColumnName() liefert den Name einer Spalte zurück passend zu der Spaltennummer (beginnend bei 1).

Beispiel 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Bitte beachten Sie auch **OCINumCols()**, **OCIColumnType()**, und **OCIColumnSize()**.

OCIColumnType (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Liefert den Datentyp einer Spalte zurück.

mixed **OCIColumnName** (int stmt, int col)

OCIColumnType() liefert den Datentyp einer Spalte zurück passend zu der Spaltennummer (beginnend bei 1).

Beispiel 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Bitte beachten Sie auch **OCINumCols()**, **OCIColumnName()**, und **OCIColumnSize()**.

OCIParse (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Analysiert eine Abfrage und gibt eine Anweisung zurück.

int **OCIParse** (int conn, strint query)

OCIParse() analysiert die *query* und benutzt dabei die *conn*. Es liefert `TRUE` zurück, wenn die Abfrage gültig ist, `FALSE`, wenn nicht. *query* kann jede Form einer gültigen SQL Anfrage sein.

OCIError (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt den letzten Fehler von stmt|conn|global zurück oder FALSE, wenn kein Fehler aufgetreten ist.

mixed **OCIError** ([array stmt|conn])

OCIError() liefert den letzten Fehler. Ist der optionale Parameter *stmt|conn* nicht angegeben, wird der zuletzt aufgetretene Fehler zurückgeliefert. Ist kein Fehler gefunden, gibt **OCIError()** FALSE zurück. **OCIError()** gibt den Fehler als assoziatives Array zurück, wobei *code* den Oracle Fehlercode und *message* den Fehlerstring beinhaltet.

OCIInternalDebug (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Aktiviert oder deaktiviert die interne Debugausgabe. Die Voreinstellung ist deaktiviert.

void **OCIInternalDebug** (int onoff)

OCIInternalDebug() aktiviert die interne Debugausgabe. *onoff* kann auf 0 gesetzt werden, um die Ausgabe abzuschalten und auf 1, um sie anzuschalten.

OCICancel (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

bricht das lesen von einem cursor ab

int **OCICancel** (int stmt)

Wenn keine weiteren Zeilen gefetched werden sollen, kann der Cursor durch **OCICancel()** abgebrochen werden.

OCISetPrefetch (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

setzt die Anzahl der Zeilen die vorab gefetcht werden sollen

int **OCISetPrefetch** (int stmt, int rows)

OCISetPrefetch() setzt sie Anzahl der Zeilen fest, welche bereits vorab gefetcht werden sollen. Der Grundwert ist 1.

OCIWriteLobToFile (PHP 4 >= 4.0.0)

Kommt noch.

void **OCIWriteLobToFile** (object lob [, string filename [, int start [, int lenght]]])

Kommt noch.

OCISaveLobFile (PHP 4 >= 4.0.0)

Kommt noch.

string **OCISaveLobFile** (object lob)

Kommt noch.

OCISaveLob (PHP 4 >= 4.0.0)

Kommt noch.

string **OCISaveLob** (object lob)

Kommt noch.

OCILoadLob (PHP 4 >= 4.0.0)

Kommt noch.

string **OCILoadLob** (object lob)

Kommt noch.

OCIColumnScale (PHP 4 >= 4.0.0)

Kommt noch.

int **OCIColumnScale** (int stmt, int col)

Kommt noch.

OCIColumnPrecision (PHP 4 >= 4.0.0)

Kommt noch.

int **OCIColumnPrecision** (int stmt, int col)

Kommt noch.

OCIColumnTypeRaw (PHP 4 >= 4.0.0)

Kommt noch.

mixed **OCIColumnTypeRaw** (int stmt, int col)

Kommt noch.

OCINewCollection (PHP 4 >= 4.0.6)

Kommt noch.

string **OCINewCollection** (int conn, string tdo [, string shema])

Kommt noch.

OCIFreeCollection (PHP 4 >= 4.1.0)

Kommt noch.

string **OCIFreeCollection** (object lob)

Kommt noch.

OCICollAssign (PHP 4 >= 4.0.6)

Kommt noch.

string **OCICollAssign** (object collection, object object)

Kommt noch.

OCICollAssignElem (PHP 4 >= 4.0.6)

Kommt noch.

string **OCICollAssignElem** (object collection, string ndx, string val)

Kommt noch.

OCICollGetElem (PHP 4 >= 4.0.6)

Kommt noch.

string **OCICollGetElem** (object collection, string ndx)

Kommt noch.

OCICollMax (PHP 4 >= 4.0.6)

Kommt noch.

string **OCICollMax** (object collection)

Kommt noch.

OCICollSize (PHP 4 >= 4.0.6)

Kommt noch.

string **OCICollSize** (object collection)

Kommt noch.

OCICollTrim (PHP 4 >= 4.0.6)

Kommt noch.

string **OCICollTrim** (object collection, int num)

Kommt noch.

LXIX. OpenSSL Funktionen

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

Einführung

Dieses Modul nutzt die Funktionen von OpenSSL (http://www.openssl.org/) zur Erzeugung und Überprüfung von Signaturen und zum versiegeln (verschlüsseln) und öffnen (entschlüsseln) von Daten. PHP-4.0.4pl1 setzt OpenSSL >= 0.9.6 voraus, PHP-4.0.5 und folgende funktionieren auch mit OpenSSL >= 0.9.5.

Anmerkung: Bitte beachten Sie, dass diese Extension immer noch als experimentell einzustufen ist!

OpenSSL bietet viele Möglichkeiten die dieses Modul zum jetzigen Zeitpunkt nicht unterstützt. Einige dieser Möglichkeiten werden vielleicht in zukünftigen Versionen zur Verfügung stehen.

Schlüssel/Zertifikat Parameter

Ziemlich viele der OpenSSL Funktionen benötigen einen Schlüssel- oder einen Zertifikatparameter. PHP 4.0.5 und frühere Versionen müssen eine Schlüssel- oder Zertifikatsresource nutzen, die von einer der openssl_get_xxx Funktionen zurückgegeben wird. Spätere Versionen können eine der folgenden Methoden benutzen:

- Zertifikate
  1. Eine X.509 Resource die von openssl_x509_read zurückgegeben wird.
  2. Eine Zeichenkette mit dem Format `file://pfad/zu/cert.pem`; die angegebene Datei muss ein PEM verschlüsseltes Zertifikat enthalten.
  3. Eine Zeichenkette die den Inhalt eines Zertifikats enthält, PEM verschlüsselt.
- Öffentliche/Private Schlüssel
  1. Eine Schlüsselresource die von openssl_get_publickey() oder openssl_get_privatekey() zurückgegeben wird.
  2. Nur für öffentliche Schlüssel: eine X.509 Resource.
  3. Eine Zeichenkette mit dem Format `file://pfad/zu/datei.pem` - die angegebene Datei muss ein PEM verschlüsseltes Zertifikat/privaten Schlüssel enthalten (kann auch beides

enthalten).

4. Eine Zeichenkette die den Inhalt eines Zertifikats/Schlüssels enthält, PEM verschlüsselt.
5. Für private Schlüssel können Sie auch die Syntax verwenden *array($key, $passphrase)*. $key repräsentiert einen Schlüssel, den Sie entweder mit der Notation file:// oder dem textlichen Inhalt wie oben angeben. $passphrase ist eine Zeichenfolge, die die Passphrase für den privaten Schlüssel enthält.

Zertifikat Nachweis

Wenn Sie eine Funktion aufrufen, die eine Signatur/Zertifikat überprüft, ist der Parameter *cainfo* ein Array, das Datei- und Verzeichnisnamen enthält, die auf vertrauenswürdige CA Dateien verweisen. Falls Sie ein Verzeichnis angeben, muss dies ein korrekt geformtes gehashedes Verzeichnis sein, in der Art, wie es der Befehl **openssl** verwenden würde.

PKCS7 Kennzeichen/Konstanten

Die S/MIME Funktionen nutzen Kennzeichen die sich dadurch auszeichnen, dass sie ein Bitfeld benutzen, das einen oder mehrere der folgenden Werte enthalten kann:

Tabelle 1. PKCS7 KONSTANTEN

Konstante	Beschreibung
PKCS7_TEXT	Fügt den text/plain Inhaltstyp Header einer verschlüsselten/signierten Nachricht hinzu. Wenn Sie entschlüsseln oder überprüfen, werden diese Header von der Ausgabe entfernt - wenn die entschlüsselte oder geprüfte Nachricht nicht dem MIME Typ text/plain entspricht, tritt ein Fehler auf.
PKCS7_BINARY	Normalerweise wird die Nachricht in ein "kanonisches" Format konvertiert, das effektiv CR und LF als Zeilende benutzt: wie von der S/MIME Spezifikation gefordert. Wird diese Option angegeben, findet keine Konvertierung statt. Das kann sich als nützlich erweisen, wenn Sie mit binären Daten umgehen, die nicht dem MIME Format entprechen.

Konstante	Beschreibung
PKCS7_NOINTERN	Wenn Sie eine Nachricht überprüfen, werden in der Nachricht enthaltene Zertifikate (falls es welche gibt) nach dem unterzeichnenden Zertifikat durchsucht. Mit dieser Option werden nur Zertifikate benutzt, die mit dem Parameter `extracerts` in der Funktion openssl_pkcs7_verify() angegeben wurden. Die mitgelieferten Zertifikate können aber dennoch als nicht vertrauenswürdige CAs benutzt werden.
PKCS7_NOVERIFY	Keine Überprüfung des signierenden Zertifikats einer signierten Nachricht.
PKCS7_NOCHAIN	Keine Verknüpfung der Überprüfung des Zertifikats mit den Zertifikaten des Unterzeichners: das bedeutet, die Zertifikate in der signierten Nachricht werden nicht als unvertrauenswürdige CAs benutzt.
PKCS7_NOCERTS	Beim Signieren einer Nachricht wird normalerweise das Zertifikat des Unterzeichnenden mit in die Nachricht aufgenommen - mit dieser Option wird das Zertifikat nicht mit aufgenommen. Diese Option verringert die Größe der signierten Nachricht, aber der Überprüfende muss eine lokale Kopie des Zertifikats vom Unterzeichnenden besitzen (zum Beispiel durch Angabe des Parameters `extracerts` in der Funktion openssl_pkcs7_verify()).
PKCS7_NOATTR	Beim Signieren einer Nachricht wird normalerweise eine Reihe von Attributen mit angehängt. Diese enthalten die Zeit der Signierung und die unterstützen symmetrischen Algorithmen. Mit dieser Option werden diese Attribute nicht mit aufgenommen.
PKCS7_DETACHED	Beim Signieren einer Nachricht wird Klartext-Signierung verwendet mit MIME Typ multipart/signed. Das ist der Standard, wenn Sie keinen optionalen Parameter `flags` an die Funktion openssl_pkcs7_sign() übergeben. Wenn Sie diese Option ausschalten wird die Nachricht mit einer undurchsichtigen Signatur unterzeichnet. Dieses Verhalten ist robuster gegenüber der Übersetzung durch Mail-Relays, kann aber nicht mit Mailprogrammen gelesen werden, die nicht den S/Mime Standard unterstützen.
PKCS7_NOSIGS	Kein Versuch und keine Überprüfung der Signaturen einer Nachricht.

Anmerkung: Diese Konstanten wurden in 4.0.6 hinzugefügt.

openssl_error_string (PHP 4 >= 4.0.6)

Liefert eine OpenSSL Fehlermeldung

mixed **openssl_error_string** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Liefert einen Fehlertext, oder `FALSE`, wenn es keine weiteren Fehler gibt.

Die Funktion **openssl_error_string()** liefert den letzten Fehler der OpenSSL Bibliothek. Die Fehlermeldungen werden auf einem Stack abgelegt. Diese Funktion sollte also mehrmals aufgerufen werden, um alle Informationen zu erhalten.

Die Parameter/Rückgabewerte dieser Funktion können sich ändern, bevor diese in einer Release Version von PHP zur Verfügung stehen.

Beispiel 1. openssl_error_string() Beispiel

```
// Annahme: Sie haben eine openssl-Funktion aufgerufen, die einen Fehler
// verursacht hat.
while($msg = openssl_error_string())
    echo $msg . "<br>";
```

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_free_key (PHP 4 >= 4.0.4)

Freigabe einer Schlüsselressource

void **openssl_free_key** (resource key_identifier)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_free_key()** löscht den Schlüssel mit dem angegebenen Parameter `key_identifier` aus dem Speicher.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Bereitet einen PEM formatierten privaten Schlüssel zum Gebrauch vor

resource **openssl_get_privatekey** (mixed key [, string passphrase])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Rückgabewert: im Erfolgsfall eine positive Schlüssel-Resource-Kennung, `FALSE` im Fehlerfall.

Die Funktion **openssl_get_privatekey()** analysiert den PEM formatierten privaten Schlüssel der mit dem Parameter `key` angegeben wurde und bereitet ihn für den Gebrauch durch andere Funktionen vor. Der optionale Parameter `passphrase` muss angegeben werden, falls der Schlüssel selbst verschlüsselt ist (geschützt durch eine Passphrase).

openssl_get_publickey (PHP 4 >= 4.0.4)

Extrahiert einen öffentlichen Schlüssel aus einem Zertifikat und bereitet ihn für den Gebrauch vor

resource **openssl_get_publickey** (mixed certificate)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Rückgabewert: im Erfolgsfall eine positive Schlüssel-Resource-Kennung, `FALSE` im Fehlerfall.

Die Funktion **openssl_get_publickey()** extrahiert den öffentlichen Schlüssel aus einem X.509 Zertifikat, angegeben durch den Parameter `certificate`, und bereitet ihn für den Gebrauch durch andere Funktionen vor.

openssl_open (PHP 4 >= 4.0.4)

Öffnet versiegelte Daten

bool **openssl_open** (string sealed_data, string open_data, string env_key, mixed priv_key_id)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Rückgabewert: `TRUE` im Erfolgsfall, `FALSE` im Fehlerfall. Im Erfolgsfall werden die geöffneten Daten im Parameter *`open_data`* zurückgegeben.

Die Funktion **openssl_open()** öffnet (entschlüsselt) *`sealed_data`* unter Verwendung des privaten Schlüsssels, der mit der Schlüsselkennung *`priv_key_id`* und dem Umschlag Schlüssel *`env_key`* verknüpft ist und füllt *`open_data`* mit den entschlüsselten Daten. Der Umschlag Schlüssel wird bei der Versiegelung erzeugt und kann ausschlieslich für einen bestimmten privaten Schlüssel verwendet werden. Mehr Informationen erhalten Sie unter der openssl_seal() Funktion.

Beispiel 1. openssl_open() Beispiel

```
// Annahme: $sealed und $env_key enthalten versiegelte Daten
// und unseren Umschlag Schlüssel; beide haben wir von dem sie
// Versiegelnden erhalten

// privaten Schlüssel aus einer Datei holen und vorbereiten
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// entschlüsseln der Daten und speichern in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "Das sind die geöffneten Daten: ", $open;
else
    echo "Öffnen der Daten nicht möglich!";

// löschen des privaten Schlüssels aus dem Speicher
openssl_free_key($pkeyid);
```

Siehe auch openssl_seal().

openssl_seal (PHP 4 >= 4.0.4)

Versiegelt (verschlüsselt) Daten

int **openssl_seal** (string data, string sealed_data, array env_keys, array pub_key_ids)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Rückgabewert: im Erolgsfall die Länge der versiegelten Daten, `FALSE` bei einem Fehler. War der Aufruf der Funktion erfolgreich, sind die versiegelten Daten in *`sealed_data`* enthalten und der Umschlag Schlüssel steht in *`env_keys`* zur Verfügung.

Die Funktion **openssl_seal()** versiegelt (verschlüsselt) *`data`* unter Verwendung von RC4 mit einem zufällig erzeugten geheimen Schlüssel. Dieser Schlüssel wird mit jedem öffentlichen Schlüssel verschlüsselt, der mit der Kennung in *`pub_key_ids`* verknüpft ist. Jeder verschlüsselte Schlüssel wird in *`env_keys`*zurückgegeben.Das bedeutet, dass Sie versiegelte Daten an mehrere Empfänger versenden können (unter der Vorraussetzung, dass Sie irgendwann die öffentlichen Schlüssel der Empfänger erhalten haben. Jeder Empfänger muss sowohl die versiegelten Daten als auch den entsprechenden Umschlag Schlüssel erhalten, der mit dem öffentlichen Schlüssel des Empfängers versiegelt wurde.

Beispiel 1. openssl_seal() Beispiel

```
// Annahme: $data enthält die zu versiegelnden Daten

// holen und vorbereiten der öffentlichen Schlüssel unserer Empfänger
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);

// Wiederholung für den zweiten Empfänger
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// versiegeln der Nachricht, nur Besitzer von $pk1 und $pk2 können
// mit Schlüsseln $sealed entschlüsseln,
// entsprechendes gilt für $ekeys[0] und $ekeys[1].
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// Löschen der Schlüssel aus dem Speicher
openssl_free_key($pk1);
openssl_free_key($pk2);
```

Siehe auch openssl_open().

openssl_sign (PHP 4 >= 4.0.4)

Erzeugen einer Unterschrift

bool **openssl_sign** (string data, string signature, mixed priv_key_id)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Rückgabewert: `TRUE` im Erfolgsfall, `FALSE` im Fehlerfall. Ist der Aufruf der Funktion erfolgreich, steht die Signatur im Parameter *`signature`* zur Verfügung.

Die Funktion **openssl_sign()** errechnet eine Unterschrift für die im Parameter *`data`* angegebenen Daten. Für das hashing der Daten wird SHA1 benutzt und anschliessend wird mit dem privaten Schlüssel verschlüsselt, angegeben mit *`priv_key_id`*. Beachten Sie, dass die Daten selbst nicht verschlüsselt werden.

Beispiel 1. openssl_sign() Beispiel

```
// Annahme: $data enthält die Daten, die signiert werden sollen.

// holen und vorbereiten des privaten Schlüssels aus einer Datei.
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// errechnen der Unterschrift
openssl_sign($data, $signature, $pkeyid);

// Löschen des Schlüssels aus dem Speicher
openssl_free_key($pkeyid);
```

Siehe auch openssl_verify().

openssl_verify (PHP 4 >= 4.0.4)

Überprüft eine Unterschrift

int **openssl_verify** (string data, string signature, mixed pub_key_id)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Rückgabewerte: Bei korrekter Unterschrift wird 1 zurückgegeben, ist die Unterschrift nicht korrekt wird 0 zurückgegeben. Tritt ein Fehler auf wird -1 zurückgegeben.

Die Funktion **openssl_verify()** überprüft die Korrektheit der Unterschrift `signature` für die angegebenen Daten `data` mit Hilfe des öffentlichen Schlüssels, verknüpft über den Parameter `pub_key_id`. Das muss der passende öffentliche zum privaten Schlüssel sein, der für die Unterschrift benutzt wurde.

Beispiel 1. openssl_verify() Beispiel

```
// Annahme: $data und $signature enthalten die Daten und die Unterschrift

// holen und vorbereiten des öffentlichen Schlüssels aus dem Zertifikat
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// feststellen, ob die Unterschrift ok ist oder nicht
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "gut";
elseif ($ok == 0)
    echo "schlecht";
else
    echo "Mist, Fehler beim Überprüfen der Unterschrift;

// Löschen des Schlüssels aus dem Speicher
openssl_free_key($pubkeyid);
```

Siehe auch openssl_sign().

openssl_pkcs7_decrypt (PHP 4 >= 4.0.6)

Entschlüssen einer S/MIME verschlüsselten Nachricht

bool **openssl_pkcs7_decrypt** (string infilename, string outfilename, mixed recipcert, mixed recipkey)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_pkcs7_decrypt()** entschlüsselt die mit S/MIME verschlüsselte Nachricht, die sich in der Datei befindet, die mit dem Paramter *infilename* angegeben wurde. Benutzt wird dafür das Zertifikat, angegeben durch *recipcert*, und der damit verknüpfte private Schlüssel, angegeben durch *recipkey*.

Die entschlüsselte Nachricht wird in die Datei geschrieben, die mit dem Parameter *outfilename* angegeben wurde.

Beispiel 1. openssl_pkcs7_decrypt() Beispiel

```
// Annahme: $cert enthält Ihr persönliches Zertifikat und
// $key Ihr privates Schlüsselpaar. Sie erhalten eine S/MIME Nachricht.

$infilename = "encrypted.msg";  // diese Datei enthält ihre verschlüsselte Nachricht
$outfilename = "decrypted.msg"; // stellen Sie sicher, dass Sie Schreibrech-
te haben!

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key))
    echo "entschlüsselt!";
else
    echo "entschlüsseln fehlgeschlagen!";
```

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_pkcs7_encrypt (PHP 4 >= 4.0.6)

Verschlüsseln einer S/MIME Nachricht

bool **openssl_pkcs7_encrypt** (string infilename, string outfilename, mixed recipcerts, array headers [, long flags])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_pkcs7_encrypt()** verschlüsselt die Inhalte der Datei, angegeben durch den Pararmeter *infilename*. Die Verschlüsselung benutzt eine RC2 40 Bit Chiffre, so dass diese Inhalte nur von den beabsichtigten Empfängern gelesen werden können, die mit dem Parameter *recipcerts* angegeben wurden. Der Parameter *recipcerts* kann ein einzelnes X.509 Zertifikat oder ein Array von X.509 Zertifikaten enthalten. Der Parameter *headers* ist ein Array aus Headern die den Daten vorangestellt wird, nachdem diese verschlüsselt wurden. Das Array *headers* kann entweder ein assoziatives Array sein, mit Indizes aus den Headernamen, oder ein indiziertes Array, wobei jedes Element eine einzelne Header-Zeile enthält. Um den Verschlüsselungsprozess zu beeinflussen, können Sie den Paramter *flags* benutzen - siehe PKCS7 Konstanten.

Beispiel 1. openssl_pkcs7_encrypt() Beispiel

```
// Die Nachricht die Sie verschlüsselt an Ihren Geheimagenten,
// bekannt als Nachtfalke, schicken wollen.
// Sein Zertifikat liegt Ihnen in der Datei nighthawk.pem vor.
$data = <<<EOD
Nachtfalke,

Streng geheim, nur für Ihre Augen bestimmt!

Der Feind kommt näher! Treffen Sie mich im Cafe um 8.30 abends.
Sie erhalten Ihren gefälschten Ausweis!

HQ
EOD;
// speichern der Nachricht in einer Datei
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// verschlüsseln
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", "nighthawk.pem",
    array("To" => "nighthawk@agent.com", // keyed syntax
          "From: HQ <hq@cia.com>", // indexed syntax
          "Subject" => "Eyes only")))
{
    // Nachricht verschlüsselt - ab damit!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
```

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_pkcs7_sign (PHP 4 >= 4.0.6)

Unterzeichnen einer S/MIME Nachricht

bool **openssl_pkcs7_sign** (string infilename, string outfilename, mixed signcert, mixed privkey, array headers [, long flags [, string extracertsfilename]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_pkcs7_sign()** unterzeichnet die Inhalte der Datei, angegeben durch den Pararmeter `infilename`. Beim Unterzeichnen wird das Zertifikat, angegeben durch `signcert`, und der dazu gehörige private Schlüssel, angegeben durch `privkey`, benutzt.

Der Parameter `headers` ist ein Array aus Headern die den Daten vorangestellt werden, nachdem diese unterzeichnet wurden (Für mehr Informationen über das Format dieses Parameters schauen Sie bei der Funktion openssl_pkcs7_encrypt() nach).

Der Parameter `flags` kann zur Änderung der Ausgabe benutzt werden - siehe PKCS7 Konstanten - falls nicht angegeben ist die Vorgabe PKCS7_DETACHED.

Der Parameter `extracertsfilename` gibt den Namen einer Datei an, die ein Bündel von extra Zertifikaten enthält, die in die Unterschrift mit eingefügt werden. Das kann zum Beispiel als Hilfe für einen Empfänger dienen, das von Ihnen benutze Zertifikat zu überprüfen.

Beispiel 1. openssl_pkcs7_sign() Beispiel

```
// Die Nachricht die Sie unterschreiben möchten, damit der Empfänger sicher sein,
// dass diese Nachricht von Ihnen stammt.

$data = <<<EOD

Sie haben meine Erlaubnis 20,000.- DM für Essen auszugeben.

Der CEO
EOD;
// Nachricht in einer Datei speichern
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// encrypt it
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("mycert.pem", "mypassphrase"),
    array("To" => "joes@sales.com", // keyed syntax
          "From: HQ <ceo@sales.com>", // indexed syntax
          "Subject" => "Eyes only"))
{
    // Nachricht unterschrieben - abschicken!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
```

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_pkcs7_verify (PHP 4 >= 4.0.6)

Überprüft die Unterschrift einer mit S/MIME unterschriebenen Nachricht

bool **openssl_pkcs7_verify** (string filename, int flags [, string outfilename [, array cainfo [, string extracerts]]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_pkcs7_verify()** liest die S/MIME Nachricht, die in der Datei `filename` enthalten ist und untersucht die digitale Unterschrift. Ist die Unterschrift überprüft wird `TRUE` zurückgegeben, falls die Unterschrift nicht korrekt ist `FALSE` (die Nachricht wurde mit dem Zertifikat vermischt oder das unterzeichnende Zertifikat ist ungültig). Tritt ein Fehler auf wird -1 zurückgegeben.

Mit dem Parameter `flags` könnenn Sie die Art und Weise beeinflussen, wie die Unterschrift überprüft wird - für mehr Informationen siehe PKCS7 Konstanten.

Wird der Parameter `outfilename` angegeben, sollte dies eine Zeichenkette sein, die den Namen einer Datei enthält, in der das Zertifikat der unterzeichnenden Person im PEM Format gespeichert wird.

Wird der Parameter `cainfo` angegeben, sollten Informationen über die vertrauenswürdigen CA Zertifikate enthalten sein , die für den Überprüfungsprozess benutzt werden - für mehr Informationen zu diesem Parameter sollten Sie unter Zertifikat Nachweis nachschauen.

Wird der Parameter `extracerts` angegeben, ist es der Name einer Datei, die ein Bündel von Zertifikaten enthält, die als nicht vertrauenswürdige CAs genutzt werden.

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_x509_checkpurpose (PHP 4 >= 4.0.6)

Überprüft ob ein Zertifikat für einen bestimmten Zweck benutzt werden kann

bool **openssl_x509_checkpurpose** (mixed x509cert, int purpose, array cainfo [, string untrustedfile])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Kann das Zertifikat für den beabsichtigten Zweck benutzt werden, gibt diese Funktion `TRUE` zurück; falls nicht `FALSE`, bei einem Fehler -1.

Die Funktion **openssl_x509_checkpurpose()** untersucht das mit dem Parameter *`x509cert`* angegebene Zertifikat darauf, ob es für den mit *`purpose`* angegebenen Zweck benutzt werden kann.

Der Parameter *`cainfo`* sollte ein Array aus vertrauenswürdigen CA Dateien/Verzeichnissen enthalten. Dieses Array ist genauer beschrieben unter Zertifikat Nachweis.

Der Parameter *`untrustedfile`*, falls angegeben, ist der Name einer PEM verschlüsselten Datei, die Zertifikate enthält, die dazu benutzt werden können, das Zertifikat zu überprüfen, obwohl kein Vertrauen zu diesen Zertifikaten besteht, die aus dieser Datei stammen.

Tabelle 1. openssl_x509_checkpurpose() Zweck

Konstante	Beschreibung
X509_PURPOSE_SSL_CLIENT	Client-Seite einer SSL Verbindung?
X509_PURPOSE_SSL_SERVER	Server-Seite einer SSL Verbindung?
X509_PURPOSE_NS_SSL_SERVER	Netscape SSL Server?
X509_PURPOSE_SMIME_SIGN	Unterschreiben einer S/MIME Email?
X509_PURPOSE_SMIME_ENCRYPT	Verschlüsseln einer S/MIME Email?
X509_PURPOSE_CRL_SIGN	Unterschreiben einer Zertifikats-Widerrufliste(CRL)?
X509_PURPOSE_ANY	Beliebige/alle Zwecke?

Diese Optionen sind keine Bitfelder - Sie können nur eine einzige angeben!

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_x509_free (PHP 4 >= 4.0.6)

Freigabe einer Zertifikats Resource

void **openssl_x509_free** (resource x509cert)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_x509_free()** löscht den Speicherbereich, der von einem Zertifikat, angegeben durch den Parameter `x509cert`, belegt wurde.

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_x509_parse (PHP 4 >= 4.0.6)

Analyse eines X509 Zertifikats und Rückgabe der Information in einem Array

array **openssl_x509_parse** (mixed x509cert [, bool shortnames])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_x509_parse()** liefert Informationen über das angegebene Zertifikat `x509cert`. Die Informationen sind: Name des Gegenstands, Name des Herausgebers, Zweck, Zeitangaben, wie gültig seit... gültig bis, usw. Der Parameter `shortnames` kontrolliert wie die Daten im Array indiziert werden - wenn `shortnames` `TRUE` ist (Standard), werden die Felder in Kurzform indiziert, ansonsten werden lange Namen verwendet - z.B.: CN ist die Kurzform von commonName.

Die Struktur der zurückgegeben Daten ist bis jetzt (absichtlich) nicht dokumentiert, da es noch vorbehaltliche Änderungen geben wird

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

openssl_x509_read (PHP 4 >= 4.0.6)

Analysiert ein X.509 Zertitifikat und gibt eine Resource-Kennung zurück

resource **openssl_x509_read** (mixed x509certdata)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **openssl_x509_read()** analysiert das Zertifikat das mit dem Parameter `x509certdata` angegeben wurde und gibt eine Resource-Kennung zurück.

Anmerkung: Diese Funktion wurde in 4.0.6 hinzugefügt.

LXX. Oracle Funktionen

Ora_Bind (PHP 3, PHP 4 >= 4.0.0)

Bindet eine PHP-Variable an einen Oracle-Parameter

int **ora_bind** (int cursor, string PHP-Variablenname, string SQL-Parametername, int Länge [, int Typ])

Gibt nach erfolgter Bindung `TRUE`, andernfalls `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion bindet die genannte PHP-Variable an einen SQL-Parameter. Der SQL-Parameter muss von der Form ":name" sein. Mit dem optionalen Parameter *Typ* können sie definieren, ob der SQL-Parameter ein in/out- (0, Grundeinstellung), in- (1) oder out- (2) Parameter ist. Ab PHP 3.0.1 können Sie statt der Zahlen die Konstanten ORA-BIND_INOUT, ORA_BIND_IN und ORA_BIND_OUT verwenden.

ora_bind() muss nach ora_parse() und vor ora_exec() aufgerufen werden. Die Eingabewerte können durch Zuweisung an die gebundenen PHP-Variablen angegeben werden. Nach dem Aufruf von ora_exec() enthalten die gebunden PHP-Variablen die Ausgabewerte, falls diese verfügbar sind.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "ergebnis", ":x", $len, 2);
ora_bind($curs, "eingabe", ":in", 5, 1);
ora_bind($curs, "ausgabe", ":out", 5, 2);
$eingabe = 765;
ora_exec($curs);
echo "Ergebnis: $ergebnis<BR>Ausgabe: $ausgabe<BR>Eingabe: $eingabe";
?>
```

Ora_Close (PHP 3, PHP 4 >= 4.0.0)

Schließt einen Oracle-Cursor

int **ora_close** (int Cursor)

Gibt nach erfolgtem Schließen `TRUE`, andernfalls `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion schließt den mittels ora_open() geöffneten Daten-Cursor.

Ora_ColumnName (PHP 3, PHP 4 >= 4.0.0)

Liefert den Namen eines Oracle-Ergebnisfeldes

string **Ora_ColumnName** (int Cursor, int Feld)

Gibt den Namen des Feldes *Feld* des Cursors *Cursor* zurück. Der zurückgegebene Name ist komplett in Großbuchstaben. Feld 0 ist das erste Feld.

Ora_ColumnSize (PHP 3, PHP 4 >= 4.0.0)

Liefert die Größe eines Oracle-Ergebnisfeldes

int **Ora_ColumnSize** (int Cursor, int Feld)

Gibt die Größe des Oracle-Feldes *Feld* des Cursors *Cursor* zurück. Feld 0 ist das erste Feld.

Ora_ColumnType (PHP 3, PHP 4 >= 4.0.0)

Liefert den Typ eines Oracle-Ergebnisfeldes

string **Ora_ColumnType** (int Cursor, int Feld)

Gibt den Oracle-Datentyp des Feldes *Feld* des Cursors *Cursor* zurück. Feld 0 ist das erste Feld. Der zurückgegebene Typ ist einer der folgenden:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit (PHP 3, PHP 4 >= 4.0.0)

Bestätigt eine Oracle-Transaktion

int **ora_commit** (int conn)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion bestätigt eine Oracle-Transaktion. Eine Transaktion ist definiert als Gesamtheit aller Änderungen seit der letzten Bestätigung/dem letzten Rollback, seit die automatische Bestätigung abgeschaltet wurde oder seit die Verbindung besteht.

Ora_CommitOff (PHP 3, PHP 4 >= 4.0.0)

Schaltet die automatische Bestätigung ab

int **ora_commitoff** (int conn)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion schaltet die automatische Bestätigung nach jedem Aufruf von ora_exec() ab.

Ora_CommitOn (PHP 3, PHP 4 >= 4.0.0)

Schaltet die automatische Bestätigung ein

int **ora_commiton** (int conn)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion schaltet die automatische Bestätigung nach jedem Aufruf von ora_exec() über die angegebene Verbindung ein.

Ora_Do (PHP 3, PHP 4 >= 4.0.0)

Kombiniert die Funktionen Ora_Parse, Ora_Exec und Ora_Fetch

int **ora_do** (int conn, string Abfrage)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion ist eine schnelle Kombination von ora_parse(), ora_exec() und ora_fetch(). Sie analysiert eine Anweisung, führt sie aus und ruft dann den ersten Datensatz des Ergebnisses ab.

Siehe auch: ora_parse(), ora_exec() und ora_fetch().

Ora_Error (PHP 3, PHP 4 >= 4.0.0)

Liefert eine Oracle-Fehlermeldung

string **Ora_Error** (int Cursor_oder_Verbindung)

Gibt eine Fehlermeldung in der Form *XXX-NNNNN* zurück, worin *XXX* angibt, wo der Fehler auftrat und *NNNNN* die Fehlermeldung identifiziert.

Anmerkung: Die Unterstützung von Verbindungs-IDs wurde in 3.0.4 eingeführt.

Bei den UNIX-Versionen von Oracle können Sie Einzelheiten über eine Fehlermeldung auf die folgende Weise finden: `$ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different`

```
level. // *Action: Either remove the unique restriction or do not insert
the key
```

Ora_ErrorCode (PHP 3, PHP 4 >= 4.0.0)

Liefert den Oracle-Fehlercode

int **Ora_ErrorCode** (int Cursor_oder_Verbindung)

Gibt den numerischen Fehlercode der zuletzt ausgeführten Anweisung des angegebenen Cursors, bzw. der angegebenen Verbindung zurück.

Anmerkung: Die Unterstützung von Verbindungs-IDs wurde in 3.0.4 eingeführt.

Ora_Exec (PHP 3, PHP 4 >= 4.0.0)

Führt eine analysierte Anweisung an einem Oracle-Cursor aus

int **ora_exec** (int Cursor)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Siehe auch: ora_parse(), ora_fetch() und ora_do().

Ora_Fetch (PHP 3, PHP 4 >= 4.0.0)

Ruft einen Datensatz eines Cursors ab

int **ora_fetch** (int Cursor)

Gibt `TRUE` (Datensatz wurde abgerufen) oder `FALSE` (keine weiteren Datensätze oder Auftreten eines Fehlers) zurück. Bei Auftreten eines Fehlers bekommen Sie die Einzelheiten mit den Funktionen ora_error() und ora_errorcode(). Wenn es keinen Fehler gab, gibt ora_errorcode() 0 zurück.

Diese Funktion ruft einen Datensatz des angegebenen Cursors ab.

Siehe auch: ora_parse(), ora_exec() und ora_do().

Ora_Fetch_Into (PHP 3, PHP 4 >= 4.0.0)

Ruft einen Datensatz in ein angegebenes Ergebnisarray ab

int **ora_fetch_into** (int Cursor, array Ergebnis [, int Flags])

Mit dieser Funktion können sie einen Datensatz in ein Array abrufen.

Beispiel 1. Oracle-Abruf in ein Array

```
<?php
array($ergebnisse);
ora_fetch_into($cursor, &$ergebnisse);
echo $ergebnisse[0];
echo $ergebnisse[1];
?>
```

Beachten Sie, dass Sie das Array als Verweis einbringen müssen.

Siehe auch: ora_parse(), ora_exec(), ora_fetch() und ora_do().

Ora_GetColumn (PHP 3, PHP 4 >= 4.0.0)

Liefert die Daten eines abgerufenen Feldes

mixed **ora_getcolumn** (int Cursor, mixed Feld)

Gibt die Felddaten zurück. Falls ein Fehler auftritt, wird `FALSE` zurückgegeben und ora_errorcode() gibt einen Wert (nicht `NULL`) zurück. Beachten Sie aber, dass die false-Überprüfung der Ergebnisse dieser Funktion auch dann wahr ergeben kann, wenn gar kein Fehler auftrat (Ergebnis `NULL`, leere Zeichenkette, die Zahl 0, die Zeichenkette "0").

Diese Funktion liefert die Daten eines Feld- oder Funktionsergebnisses.

Ora_Logoff (PHP 3, PHP 4 >= 4.0.0)

Beendet eine Oracle-Verbindung

int **ora_logoff** (int Verbindung)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion meldet den Benutzer ab und trennt die Verbindung zum Server.

See also ora_logon().

Ora_Logon (PHP 3, PHP 4 >= 4.0.0)

Baut eine Oracle-Verbindung auf

int **ora_logon** (string Benutzername, string Passwort)

Gibt bei Erfolg einen Verbindungs-Index oder bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion baut mit *`Benutzername`* und *`Passwort`* eine Verbindung zwischen PHP und einer Oracle Datenbank auf.

Verbindungen können unter Verwendung von SQL*Net durch Eingabe des TNS-Namens zu *`Benutzername`* wie folgt aufgebaut werden:

```
$conn = Ora_Logon("Benutzername<emphasis>@TNSNAME</emphasis>", "pass");
```

Falls Sie einen Zeichensatz mit nicht-ASCII-Zeichen verwenden, sollten Sie sicherstellen, dass in ihrer Umgebung NLS_LANG gesetzt ist. Bei Servermodulen sollten Sie sie in der Umgebung des Servers setzen, bevor Sie den Server starten.

Ora_pLogon (PHP 3, PHP 4 >= 4.0.0)

Baut eine persistente Oracle-Verbindung auf

int **ora_plogon** (string Benutzername, string Passwort)

Diese Funktion baut mit *`Benutzername`* und *`Passwort`* eine persistente Verbindung zwischen PHP und einer Oracle Datenbank auf.

Siehe auch: ora_logon().

Ora_Numcols (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Felder

int **ora_numcols** (int cursor_ind)

ora_numcols() gibt die Anzahl der Felder eines Ergebnisses zurück. Aussagekräftige Werte werden nur nach einer analysieren/ausführen/abrufen-Abfolge zurückgegeben.

Siehe auch: ora_parse(), ora_exec(), ora_fetch() und ora_do().

Ora_Numrows (PHP 3, PHP 4 >= 4.0.0)

Liefert die Anzahl der Datensätze

int **ora_numrows** (int cursor_ind)

ora_numrows() gibt die Anzahl der Datensätze eines Ergebnisses zurück.

Ora_Open (PHP 3, PHP 4 >= 4.0.0)

Öffnet einen Oracle-Cursor

int **ora_open** (int Verbindung)

Gibt bei Erfolg einen Cursor-Index oder bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion öffnet einen Oracle-Cursor, der mit *`Verbindung`* verknüpft ist.

Ora_Parse (PHP 3, PHP 4 >= 4.0.0)

Analysiert eine SQL-Anweisung

int **ora_parse** (int Cursor_ind, string SQL-Anweisung, int aufschieben)

Gibt bei Erfolg `TRUE` zurück, im Fehlerfall `FALSE`.

Diese Funktion analysiert eine SQL-Anweisung oder einen PL/SQL-Block und verknüpft diese(n) mit dem angegebenen Cursor.

Siehe auch: ora_exec(), ora_fetch() und ora_do().

Ora_Rollback (PHP 3, PHP 4 >= 4.0.0)

Macht eine Transaktion rückgängig

int **ora_rollback** (int Verbindung)

Gibt bei Erfolg `TRUE` und bei Auftreten eines Fehlers `FALSE` zurück. Einzelheiten über den Fehler bekommen Sie mit den Funktionen ora_error() und ora_errorcode().

Diese Funktion macht eine Oracle-Transaktion rückgängig (Definition einer Transaktion: siehe ora_commit()).

LXXI. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com (http://www.ovrimos.com/). To enable ovrimos support in PHP just compile php with the '--with-ovrimos' parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Beispiel 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to SQL Server.

ovrimos_connect (PHP 4 >= 4.0.3)

Connect to the specified database

int **ovrimos_connect** (string host, string db, string user, string password)

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Beispiel 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_close (PHP 4 >= 4.0.3)

Closes the connection to ovrimos

void **ovrimos_close** (int connection)

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Specifies how many bytes are to be retrieved from long datatypes

int **ovrimos_longreadlen** (int result_id, int length)

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. It currently sets this parameter the specified result set. Returns `TRUE`.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prepares an SQL statement

int **ovrimos_prepare** (int connection_id, string query)

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or `FALSE` on failure).

Beispiel 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_execute (PHP 4 >= 4.0.3)

Executes a prepared SQL statement

bool **ovrimos_execute** (int result_id [, array parameters_array])

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns TRUE or FALSE. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don't follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn't bring myself on liking it.

ovrimos_cursor (PHP 4 >= 4.0.3)

Returns the name of the cursor

int **ovrimos_cursor** (int result_id)

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec (PHP 4 >= 4.0.3)

Executes an SQL statement

int **ovrimos_exec** (int connection_id, string query)

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or FALSE. Evidently, the SQL statement should not contain parameters.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Fetches a row from the result set

bool **ovrimos_fetch_into** (int result_id, array result_array [, string how [, int rownumber]])

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns TRUE or FALSE.

Beispiel 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
```

```
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Fetches a row from the result set

bool **ovrimos_fetch_row** (int result_id [, int how [, int row_number]])

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns `TRUE` or `FALSE`.

Beispiel 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
```

```
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_result (PHP 4 >= 4.0.3)

Retrieves the output column

int **ovrimos_result** (int result_id, mixed field)

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by 'field', either as a string or as an 1-based index.

ovrimos_result_all (PHP 4 >= 4.0.3)

Prints the whole result set as an HTML table

bool **ovrimos_result_all** (int result_id [, string format])

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns `TRUE` or `FALSE`.

Beispiel 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
```

```
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Beispiel 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i <= $colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

Beispiel 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
```

```
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

ovrimos_num_rows (PHP 4 >= 4.0.3)

Returns the number of rows affected by update operations

int **ovrimos_num_rows** (int result_id)

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_num_fields (PHP 4 >= 4.0.3)

Returns the number of columns

int **ovrimos_num_fields** (int result_id)

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_field_name (PHP 4 >= 4.0.3)

Returns the output column name

int **ovrimos_field_name** (int result_id, int field_number)

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_type (PHP 4 >= 4.0.3)

Returns the (numeric) type of the output column

int **ovrimos_field_type** (int result_id, int field_number)

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_field_len (PHP 4 >= 4.0.3)

Returns the length of the output column

int **ovrimos_field_len** (int result_id, int field_number)

ovrimos_field_len() is used to get the length of the output column with number *field_number*, in result *result_id*.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_num (PHP 4 >= 4.0.3)

Returns the (1-based) index of the output column

int **ovrimos_field_num** (int result_id, string field_name)

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or `FALSE`.

ovrimos_free_result (PHP 4 >= 4.0.3)

Frees the specified result_id

bool **ovrimos_free_result** (int result_id)

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id *result_id*. Returns `TRUE`.

ovrimos_commit (PHP 4 >= 4.0.3)

Commits the transaction

int **ovrimos_commit** (int connection_id)

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_rollback (PHP 4 >= 4.0.3)

Rolls back the transaction

int **ovrimos_rollback** (int connection_id)

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LXXII. Funktionen zur Ausgabesteuerung

Mit den Funktionen zur Ausgabesteuerung können Sie die Ausgabe eines Skripts steuern. Dies kann in verschiedenen Situationen ganz nützlich sein, besonders, wenn Sie header an den Browser schicken müssen, nachdem Ihr Skript bereits etwas ausgegeben hat. Die Ausgabesteuerungsfunktionen gelten nicht für header, die mit header() oder setcookie() geschickt wurden, sondern nur für Daten ,die mit echo() gesendet wurden oder für Daten zwischen PHP-Codeblöcken.

Beispiel 1. Beispiel Ausgabesteuerung

```
<?php

ob_start();
echo "Hello\n";

setcookie ("cookiename", "cookiedata");

ob_end_flush();

?>
```

Im obigen Beispiel wird die Ausgabe der echo() solange im Ausgabepuffer zwischengespeichert, bis die Funktion ob_end_flush() aufgeufen wird. In der Zwischenzeit speichert der Aufruf von setcookie() erfolgreich einen Cookie, ohne einen Fehler zu erzeugen. (Normalerweise können Sie keine header an den Browser mehr schicken, wenn schon Daten ausgegeben wurden.)

Siehe auch header() und setcookie().

flush (PHP 3, PHP 4 >= 4.0.0)

Der Ausgabe-Puffer wird geleert und alle Zeichen ausgegeben

void **flush** (void)

Unabhängig davon, wie PHP läuft (CGI oder WEB-Server), wird der gesamte Ausgabe-Puffer geleert. Dabei wird versucht den ganzen Inhalt an den Browser des Benutzers zu schicken.

ob_start (PHP 4 >= 4.0.0)

Einschalten der Pufferung von Ausgaben

void **ob_start** (void)

Diese Funktion schaltet die Pufferung von Ausgaben ein. Solange die Ausgabe-Pufferung aktiv ist, erfolgt keine Ausgabe bei der Skript-Ausführung. Statt dessen werden alle Ausgaben in einem internen Puffer gespeichert.

Der Inhalt diese internen Puffers kann in eine String-Variable kopiert werden (per ob_get_contents()). Die Ausgabe erfolgt erst bei Aufruf von ob_end_flush() und sobald der Puffer-Inhalt per ob_end_clean() verworfen wird.

Siehe auch ob_get_contents(), ob_end_flush(), ob_end_clean() und ob_implicit_flush().

ob_get_contents (PHP 4 >= 4.0.0)

Gibt den Inhalt des Ausgabe-Puffers zurück

string **ob_get_contents** (void)

Damit wird der Inhalt des Ausgabe-Puffers zurück gegeben (oder `FALSE`, wenn die Ausgabe-Pufferung nicht aktiv ist).

Siehe auch ob_start(), ob_end_flush() und ob_end_clean().

ob_end_flush (PHP 4 >= 4.0.0)

Leert (schickt/sendet) den Ausgabe-Puffer und deaktiviert die Ausgabe-Pufferung

void **ob_end_flush** (void)

Der Inhalt des Ausgabe-Puffers (sofern vorhanden) wird abgeschickt und die Pufferung der Ausgabe deaktiviert. Falls sie mit dem Puffer-Inhalt weiter arbeiten möchten, müssen sie diesen erst per ob_get_contents() zwischen speichern bevor sie **ob_end_flush()** aufrufen, da dadurch der Puffer geleert wird.

Siehe auch ob_start(), ob_get_contents() und ob_end_clean().

ob_end_clean (PHP 4 >= 4.0.0)

Bereinigt (löscht) den Ausgabe-Puffer und deaktiviert die Ausgabe-Pufferung

void **ob_end_clean** (void)

Diese Funktion verwirft den Inhalt des Ausgabe-Puffers und deaktiviert die Pufferung.

Siehe auch ob_start() und ob_end_flush().

ob_implicit_flush (PHP 4 >= 4.0.0)

Schaltet die implizite Ausgabe ein bzw. aus

void **ob_implicit_flush** ([int flag])

ob_implicit_flush() schaltet die implizite Ausgabe an oder aus (wenn kein *flag* angegeben wurde, wird per default "Ein" angenommen). Die implizite Puffer-Ausgabe erzeugt eine Ausgabe nach jedem Ausgabe-Befehl, so dass keine Extra-Aufrufe von flush() mehr erforderlich sind.

Das Einschalten der impliziten Ausgabe beendet die Pufferung der Ausgabe. Der aktuelle Puffer-Inhalt wird ausgegeben, so als ob die Funktion ob_end_flush() aufgerufen wurde.

Siehe auch flush(), ob_start() und ob_end_flush().

LXXIII. Object property and method call overloading

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

The purpose of this extension is to allow overloading of object property access and method calls. Only one function is defined in this extension, overload() which takes the name of the class that should have this functionality enabled. The class named has to define appropriate methods if it wants to have this functionality: `__get()`, `__set()` and `__call()` respectively for getting/setting a property, or calling a method. This way overloading can be selective. Inside these handler functions the overloading is disabled so you can access object properties normally.

Some simple examples on using the overload() function:

Beispiel 1. Overloading a PHP class

```
<?php

class OO
{
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // Callback method for getting a property
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // Callback method for setting a property
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// Here we overload the OO object
overload('OO');

$o = new OO;
print "\$o->a: $o->a\n"; // print: $o->a:
print "\$o->b: $o->b\n"; // print: $o->b: 9
```

```
print "\$o->c: $o->c\n"; // print: $o->c: 42
print "\$o->d: $o->d\n"; // print: $o->d:

// add a new item to the $elem array in OO
$o->x = 56;

// instantiate stdclass (it is built-in in PHP 4)
// $val is not overloaded!
$val = new stdclass;
$val->prop = 555;

// Set "a" to be an array with the $val object in it
// But __set() will put this in the $elem array
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

Warnung

As this is an experimental extension, not all things work. There is no `__call()` support currently, you can only overload the get and set operations for properties. You cannot invoke the original overloading handlers of the class, and `__set()` only works to one level of property access.

overload (PHP 4 CVS only)

Enable property and method call overloading for a class

void **overload** ([string class_name])

The **overload()** function will enable property and method call overloading for a class identified by `class_name`. See an example in the introductory section of this part.

LXXIV. PDF Funktionen

Einleitung

Mit Hilfe der PDF-Bibliothek von Thomas Merz können mit PHP PDF Dateien erzeugt werden. Die Bibliothek ist unter http://www.pdflib.com/pdflib/index.html; verfügbar. Zudem sind noch zwei weitere Bibliotheken notwendig, die JPEG Bibliothek (ftp://ftp.uu.net/graphics/jpeg/) und die TIFF Bibliothek (http://www.libtiff.org/), um PHP zu übersetzen. Diese beiden Bibliothek bereiten häufig Probleme bei der Konfiguration von PHP. Sie sollten unbedingt die Meldungen des Konfigurations-Skripts befolgen, um aufkommende Probleme zu lösen.

Beachten sie zudem die excellente Dokumentation die mit pdflib ausgeliefert wird. Sie gibt einen guten Überblick über die Möglichkeiten von pdflib.

Die meisten Funktionen in pdflib sind in ähnlicher Form auch in PHP vorhanden. Die Parameter sind in der Regel auch identisch. Sie sollten zudem die Grundkonzepte von PDF und Postscript verstanden haben, um das PDF Module effizient nutzen zu können.

Alle Längen und Koordinatenangaben sind in Postscript-Punkten gemessen. Für gewöhnlich entsprechen 72 PostScript-Punkte 1 Inch, was jedoch von der Auflösung des Ausgabegeräts abhängt.

Neben diesem Modul gibt es noch ein weiteres zur Erzeugung von PDF-Dateien basierend auf der Bibliothek ClibPDF von FastIO (http://www.fastio.com/). Es hat eine geringfügig andere Programmierschnittstelle. Schauen Sie in die Funktionsübersicht für weitere Details.

Das PDF Module von PHP verwendet einen neuen Type von Variable Er wird `pdfdoc` genannt. `pdfdoc` ist ein Zeiger auf das PDF Dokument und wird bei fast allen Funktion als erster Parameter erwartet.

Altlasten

Since the very begining of PDF support in PHP — starting with pdflib 0.6 — there has been tons of changes especially to the pdflib API. Most of these changes has been somehow covered by PHP, some has even required changes to the PHP API. Since pdflib 3.x the API seems to be stabilzed and PHP4 has adopted the version as a minimum requirement for PDF support. The consequence will be that many functions will disappear or be replaced by alternatives sooner or later. Support for pdflib 0.6 is already completely given up. The following table list all the functions which are deprecated in PHP 4.02 and should be replaced by their new versions.

Tabelle 1. Deprecated functions and its replacements

Old function	Replacement
pdf_put_image()	Not needed anymore.
pdf_get_font()	pdf_get_value() passing `"font"` as the second parameter.
pdf_get_fontsize()	pdf_get_value() passing `"fontsize"` as the second parameter.
pdf_get_fontname()	pdf_get_parameter() passing `"fontname"` as the second parameter.

Old function	Replacement
pdf_set_info_creator()	pdf_set_info() passing `"Creator"` as the second parameter.
pdf_set_info_title()	pdf_set_info() passing `"Title"` as the second parameter.
pdf_set_info_subject()	pdf_set_info() passing `"Subject"` as the second parameter.
pdf_set_info_author()	pdf_set_info() passing `"Author"` as the second parameter.
pdf_set_info_keywords()	pdf_set_info() passing `"Keywords"` as the second parameter.
pdf_set_leading()	pdf_set_value() passing `"leading"` as the second parameter.
pdf_set_text_rendering()	pdf_set_value() passing `"textrendering"` as the second parameter.
pdf_set_text_rise()	pdf_set_value() passing `"textrise"` as the second parameter.
pdf_set_horiz_scaling()	pdf_set_value() passing `"horizscaling"` as the second parameter.
pdf_set_text_matrix()	Not available anymore
pdf_set_char_spacing()	pdf_set_value() passing `"charspacing"` as the second parameter.
pdf_set_word_spacing()	pdf_set_value() passing `"wordspacing"` as the second parameter.
pdf_set_transition()	pdf_set_parameter() passing `"transition"` as the second parameter.
pdf_set_duration()	pdf_set_value() passing `"duration"` as the second parameter.
pdf_open_gif()	pdf_open_image_file() passing `"gif"` as the second parameter.
pdf_open_jpeg()	pdf_open_image_file() passing `"jpeg"` as the second parameter.
pdf_open_tiff()	pdf_open_image_file() passing `"tiff"` as the second parameter.
pdf_open_png()	pdf_open_image_file() passing `"png"` as the second parameter.
pdf_get_imagewidth()	pdf_get_value() passing `"imagewidth"` as the second parameter and the image as the third parameter.
pdf_get_imageheight()	pdf_get_value() passing `"imageheight"` as the second parameter and the image as the third parameter.
()	**()**

Tips zur Installation von pdflib 3.x

Seit Version 3.0 von pdflib sollten sie pdflib mit der configure-Option `--enable-shared-pdflib` konfigurieren.

Verwenden alter pdflib Versionen

Mit Version 2.20 von pdflib wurden einige Veränderungen an der API vorgenommen. Zudem ist Unterstützung für asiatische Zeichensätze hinzugekommen. Dies hat leider auch zu Änderungen am php4-Modul geführt (nicht php3). Wenn Sie pdflib 2.20 benutzen, dann sollten Sie die Erzeugung von Dokumenten im Speicher mit Vorsicht benutzen. Bis zur entgültigen Version 3.0 von pdflib könnte dies instabil sein. Der encoding Parameter der Funktion pdf_set_font() hat sich in eine Zeichenkette geändert. Dies bedeutet, dass anstatt von beispielsweise 4 jetzt 'winansi' verwendet werden muss.

Wenn Sie die Version 2.30 von pdflib verwenden, dann steht Ihnen die Funktion pdf_set_text_matrix() nicht mehr zur Verfügung. Diese Funktion wurde komplett entfernt. Grundsätzlich ist es ratsam die release notes der verwendeten Version von pdflib zu lesen.

Versionen von PHP4 die nach dem 9. März 2000 erscheinen unterstützen nur noch die Versionen >3.0 von pdflib. PHP3 sollte hingegen nicht mit Versionen >2.01 verwendet werden.

Bei Gebrauch von pdflib 2.01 sollte überprüft werden, ob die Bibliothek richtig installiert wurde. Es sollte die Datei oder ein Verweis libpdf.so im Installationsverzeichnis von pdflib existieren. Version 2.01 erzeugt nur die Bibliothek mit dem Namen libpdf2.01.so, die so nicht vom Linker des Systems gefunden werden kann. In diesem Fall müssen Sie den Verweis von libpdf.so nach libpdf2.01.so selbst anlegen.

Beispiele

Die meisten Funktionen sind sehr einfach zu benutzen. Das Schwierigste wird wohl sein, überhaupt ein einfaches PDF-Dokument zu erstellen. Das folgende Beispiel soll die ersten Schritte erleichtern. Dieses Skript erstellt die PDF-Datei `test.pdf`, welche nur aus einer Seite besteht. Auf der Seite befindet sich der Text "Times Roman outlined" in einem outlined, 30pt Zeichensatz. Der Text ist zudem unterstrichen.

Beispiel 1. Erstellung eines PDF Dokuments mit pdflib

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = pdf_open($fp);
pdf_set_info_author($pdf, "Uwe Steinmann");
pdf_set_info_title($pdf, "Test for PHP wrapper of PDFlib 2.0");
pdf_set_info_author($pdf, "Name of Author");
pdf_set_info_creator($pdf, "See Author");
pdf_set_info_subject($pdf, "Testing");
pdf_begin_page($pdf, 595, 842);
```

```
pdf_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php>finished</A>";
?>
```

Das PHP-Skript `getpdf.php` liefert nur das PDF-Dokument.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Die pdflib-Distribution enthält ein komplizierteres Beispiel, welches ein Reihe Seiten erstellt die jeweils eine analoge Uhr mit der aktuellen Uhrzeit enthalten. Dieses Beispiel in ein PHP-Skript umgesetzt sieht wie folgt aus (das gleiche Beispiel wird auch in der Dokumentation zum cpdf Modul verwendet):

Beispiel 2. pdfclock Beispiel aus der pdflib Distribution

```
<?php
$pdffilename = "clock.pdf";
$radius = 200;
$margin = 20;
$pagecount = 40;

$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);
pdf_set_info_creator($pdf, "pdf_clock.php3");
pdf_set_info_author($pdf, "Uwe Steinmann");
pdf_set_info_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_transition($pdf, 4);  /* wipe */
    pdf_set_duration($pdf, 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
```

```
/* minute strokes */
pdf_setlinewidth($pdf, 2.0);
for ($alpha = 0; $alpha < 360; $alpha += 6) {
    pdf_rotate($pdf, 6.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin/3, 0.0);
    pdf_stroke($pdf);
}

pdf_restore($pdf);
pdf_save($pdf);

/* 5 minute strokes */
pdf_setlinewidth($pdf, 3.0);
for ($alpha = 0; $alpha < 360; $alpha += 30) {
    pdf_rotate($pdf, 30.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin, 0.0);
    pdf_stroke($pdf);
}

$ltime = getdate();

/* draw hour hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius/2, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw minute hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius * 0.8, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw second hand */
pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
pdf_setlinewidth($pdf, 2);
pdf_save($pdf);
pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
pdf_moveto($pdf, -$radius/5, 0.0);
pdf_lineto($pdf, $radius, 0.0);
pdf_stroke($pdf);
pdf_restore($pdf);

/* draw little circle at center */
pdf_circle($pdf, 0, 0, $radius/30);
pdf_fill($pdf);
```

```
    pdf_restore($pdf);

    pdf_end_page($pdf);
}

$pdf = pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php?filename=".$pdffilename.">finished</A>";
?>
```

Das PHP-Skript `getpdf.php` liefert wiederum nur das Dokument.

```
<?php
$fp = fopen($filename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

pdf_set_info (PHP 4)

Setzt Info-Feld der Dokumenteninformation

void **pdf_set_info** (int PDF Dokument, string Feldname, string Wert)

Die Funktion **pdf_set_info()** setzt ein Feld der Dokumenteninformation. Mögliche Informationsfelder die gesetzt werden können sind 'Subject', 'Title', 'Creator', 'Author', 'Keywords' sowie ein Feld mit einem beliebigen Namen. Die Funktion kann vor dem Beginnen einer Seite aufgerufen werden.

Beispiel 1. Dokumenteninformation setzen

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

Anmerkung: Diese Funktion ersetzt **pdf_set_info_keywords()**, **pdf_set_info_title()**, **pdf_set_info_subject()**, **pdf_set_info_creator()**, **pdf_set_info_sybject()**.

pdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Öffnet ein neues PDF-Dokument

int **pdf_open** (int Datei-Deskriptor)

Die Funktion **pdf_open()** öffnet ein neues PDF-Dokument. Die korropondierende Datei muss zuvor mit mit fopen() geöffnet worden sein und der Dateizeiger als Parameter *`Datei-Deskriptor`* übergeben werden. Wenn keine Datei-Deskriptor übergeben wird, erfolgt die Erstellung des Dokuments im Speicher, und die Ausgabe geschieht seitenweise nach stdout bzw. zum Web-Browser.

Anmerkung: Der Rückgabewert dieser Funktion wird bei allen anderen Funktionen als ersten Parameter erwartet.

Siehe auch fopen(), pdf_close().

pdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schließt ein PDF-Dokument

void **pdf_close** (int PDF-Dokument)

Die **pdf_close()** Function schließt ein PDF-Dokument.

Siehe auch pdf_open(), fclose().

pdf_begin_page (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Beginnt eine neue Seite

void **pdf_begin_page** (int PDF-Dokument, double Breite, double Höhe)

Die Funktion **pdf_begin_page()** beginnt eine neue Seite mit Höhe *Höhe* und mit der Breite *Breite*. Um ein gültiges Dokument zu erzeugen muss mindestens eine Seite erzeugt werden, die mit pdf_end_page() beendet wird.

Siehe auch pdf_end_page().

pdf_end_page (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Beendet eine Seite

void **pdf_end_page** (int PDF-Dokument)

Die Funktion **pdf_end_page()** beended eine Seite. Sobald eine Seite beendet ist können keine Veränderungen an ihr mehr vorgenommen werden.

Siehe auch pdf_begin_page().

pdf_show (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schreibt einen Text an die aktuelle Position

void **pdf_show** (int PDF-Dokument, string Text)

Die Funktion **pdf_show()** gibt den Text *Text* an der aktuellen Textposition in dem aktuellen Zeichensatz aus.

Siehe auch pdf_show_xy(), pdf_show_boxed(), pdf_set_text_pos(), pdf_set_font().

pdf_show_boxed (PHP 4 >= 4.0.0)

Schreibt einen Text in eine Box

void **pdf_show_boxed** (int PDF-Dokument, string text, double x-Koor, double y-Koor, double Breite, double Höhe, string Modus [, string Feature])

Die Funktion **pdf_show_boxed()** gibt den Text *text* in einer Box aus, dessen linke untere Ecke an der Position (*x-coor*, *y-coor*) ist. Die Größe der Box ist *Höhe* mal *Breite*. Der Parameter *Modus* gibt an wie der Text gesetzt wird. Wenn *Höhe* und *Breite* `NULL` sind, darf *Modus* auf "left", "right" oder "center" gesetzt werden. Wenn *Höhe* oder *Breite* ungleich `NULL` sind zusätzlich die Werte "justify" and "fulljustify" erlaubt.

Wird im Parameter *Feature* der Wert "blind" übergeben, wird der Text nicht auszugeben.

Die Funktion liefert die Zahl der Zeichen, die nicht mehr berücksichtigt wurden, weil sie nicht mehr in Box passten.

See also pdf_show(), pdf_show_xy().

pdf_show_xy (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schreibt einen Text an die angegebene Position

void **pdf_show_xy** (int PDF-Dokument, string Text, double x-Koor, double y-Koor)

Die Funktion **pdf_show_xy()** gibt den Text *Text* an der Position (*x-Koor*, *y-Koor*) aus.

Siehe auch pdf_show(), pdf_show_boxed().

pdf_set_font (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Wählt einen Zeichensatz und dessen Größe aus

void **pdf_set_font** (int PDF-Dokument, string Zeichensatzname, double Größe, string Kodierung [, int einbetten])

Die Funktion **pdf_set_font()** setzt den den aktuellen Zeichensatz, die Zeichensatzgröße und die Kodierung. Bei Verwendung von pdflib 0.6 müssen die Adobe Zeichensatz-Metriken (afm-Dateien) für den Zeichensatz in dem voreingestellten Verzeichnis abgelegt werden (voreingestellt ist ./font).

Der vorletzte Parameter *encoding* gibt die Kodierung des Zeichensatzes an. Er kann die folgenden Werte annehmen: 'builtin', 'ebcdic', 'macroman', 'macexpert', 'winansi', 'host'. 'winansi' oder 'host' ist im übrigen häufig eine gute Wahl. Wenn der letzte Parameter auf eins gesetzt ist, wird der Zeichensatz in das PDF-Dokument eingebettet, sonst nicht. Einen Zeichensatz einzubetten, ist immer dann eine gute Wahl, wenn der Zeichensatz nicht sehr weit verbreitet ist, und die Person, die das Dokument lesen soll, keinen Zugriff auf den Zeichensatz hat.

Anmerkung: Diese Funktion muss nach pdf_begin_page() aufgerufen werden um ein gültiges Dokument zu erhalten.

Anmerkung: Wenn der Zeichensatz über eine .upr-Datei referenziert wird, muss der Name in der afm-Datei mit dem Zeichensatznamen übereinstimmen. Andernfalls wird der Zeichensatz mit jedem Aufruf von **pdf_set_font()** erneut eingebettet. (Danke an Paul Haddon, der dieses Problem gefunden hat.)

pdf_set_leading (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Abstand zwischen zwei Textzeilen

void **pdf_set leading** (int PDF-Dokument, double Abstand)

Die Funktion **pdf_set_leading()** setzt denn Abstand zwischen zwei Textzeilen, die mit pdf_continue_text() ausgegeben werden.

Siehe auch pdf_continue_text().

pdf_set_parameter (PHP 4 >= 4.0.0)

Setzt verschiedene Parameter

void **pdf_set_parameter** (int PDF-Dokument, string Name, string Wert)

Die **pdf_set_parameter()** Funktion setzt verschiedene Parameter von pdflib, die vom Typ string sind.

Siehe auch pdf_get_value(), pdf_set_value(), pdf_get_parameter().

pdf_get_parameter (PHP 4)

Liefert verschiedene Parameter

string **pdf_get_parameter** (int PDF-Dokument, string Name, double Modifizierer)

Die pdf_set_parameter() Funktion liefert verschiedene Parameter von pdflib, die vom Typ string sind. Der Funktionsparameter `Modifizierer` spezifiziert den ausgewählten Parameter genauer.

Siehe auch pdf_get_value(), pdf_set_value(), pdf_set_parameter().

pdf_set_value (PHP 4)

Setzt verschiedene numerische Werte

void **pdf_set_value** (int PDF-Dokument, string Name, double Wert)

Die **pdf_set_value()** Funktion setzt verschiedene numerische Parameter von pdflib.

Siehe auch pdf_get_value(), pdf_set_parameter(), pdf_get_parameter().

pdf_get_value (PHP 4)

Liefert verschiedene numerische Werte

double **pdf_get_value** (int PDF-Dokument, string Name, double Modifizierer)

Die **pdf_get_value()** Funktion liefert verschiedene numerische Parameter von pdflib. Der Funktionsparameter `Modifizierer` spezifiziert den ausgewählten Parameter genauer.

Siehe auch **pdf_get_value()**, pdf_set_value(), pdf_set_parameter().

pdf_get_image_height (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Liefert die Höhe eines Bildes

string **pdf_get_image_height** (int pdf document, int Bild)

Die Funktion **pdf_get_image_height()** liefert die Höhe eines Bildes in Pixel.

See also pdf_open_image_file(), pdf_open_memory_image(), pdf_get_image_width().

pdf_get_image_width (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Liefert die Breite eines Bildes

string **pdf_get_image_width** (int pdf document, int Bild)

Die Funktion **pdf_get_image_width()** liefert die Breite eines Bildes in Pixel.

See also pdf_open_image_file(), pdf_open_memory_image(), pdf_get_image_height().

pdf_set_text_rendering (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Bestimmt wie der Text ausgegeben werden soll

void **pdf_set_text_rendering** (int PDF-Dokument, int Modus)

Die Funktion **pdf_set_text_rendering()** bestimmt wie die Zeichen einer Textausgabe dargestellt werden. Die möglichen Werte für den Parameter `Modus` sind 0=ausgefüllt, 1=nur die Umrandung, 2=ausgefüllt und die Umrandung, 3=unsichtbar, 4=ausgefüllt und Hinzunahme zum Begrenzungspfad (cliping path), 5=Umrandung und Hinzunahme zum Begrenzungspfad, 6=ausgefüllt, umrandet und Hinzunahme zum Begrenzungspfad, 7=nur Hinzunahme zum Begrenzungspfad.

pdf_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die horizontale Skalierung bei der Textausgabe

void **pdf_set_horiz_scaling** (int PDF-Dokument, double Skalierung)

Die Funktion **pdf_set_horiz_scaling()** setzt den horizontalen Skalierungsfaktor auf *`Skalierung`* Prozent.

pdf_set_text_rise (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Textverschiebung

void **pdf_set_text_rise** (int PDF-Dokument, double Verschiebung)

Die Funktion **pdf_set_text_rise()** setzt die Textverschiebung nach oben auf *`Verschiebung`* Punkte.

pdf_set_text_matrix (PHP 3>= 3.0.6)

Setzt die Text-Matrix

void **pdf_set_text_matrix** (int PDF-Dokument, array Matrix)

Die Funktion **pdf_set_text_matrix()** setzt die Matrix, welche die Transformation auf Textausgaben beschreibt. text font. Die Matrix muss als array mit sechs Parametern übergeben werden.

pdf_set_text_pos (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Textposition

void **pdf_set_text_pos** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **pdf_set_text_pos()** setzt die Position für die nächste Textausgabe mit der Funktion pdf_show().

Siehe auch pdf_show(), pdf_show_xy().

pdf_set_char_spacing (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Abstand zwischen Zeichen

void **pdf_set_char_spacing** (int PDF-Dokument, double Abstand)

Die Funktion **pdf_set_char_spacing()** setzt den Abstand zwischen zwei Zeichen.

Siehe auch pdf_set_word_spacing(), pdf_set_leading().

pdf_set_word_spacing (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Abstand zwischen Wörtern

void **pdf_set_word_spacing** (int PDF-Dokument, double space)

Die Funktion **pdf_set_word_spacing()** setzt den Abstand zwischen zwei Wörtern.

Siehe auch pdf_set_char_spacing(), pdf_set_leading().

pdf_skew (PHP 4 >= 4.0.0)

Schert das Koordinatensystem

void **pdf_skew** (int PDF-Dokument, double alpha, double beta)

Die **pdf_skew()** Funktion schert das Koordinatensystem um *alpha* (x) und *beta* (y) Grad. *alpha* und *beta* dürfen weder 90 noch 270 Grad sein.

pdf_continue_text (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schreibt den Text in die nächste Zeile

void **pdf_continue_text** (int PDF-Dokument, string Text)

Die Funktion **pdf_continue_text()** gibt den Text *text* in der nächsten Zeile aus. Der Abstand zwischen zwei Zeilen kann mit der Funktion pdf_set_leading() gesetzt werden.

Siehe auch pdf_show_xy(), pdf_set_leading(), pdf_set_text_pos().

pdf_stringwidth (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert die benötigte Breite einer Zeichenkette mit dem aktuelle Zeichensatz

double **pdf_stringwidth** (int PDF-Dokument, string Text)

Die Funktion **pdf_stringwidth()** liefert die Breite eines Textes *text* unter Verwendung des aktuellen Zeichensatzes. Um diese Funktion benutzen zu können muss zuvor ein Zeichensatz mit pdf_set_font() gesetzt worden sein.

Siehe auch pdf_set_font().

pdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sichert die aktuelle Umgebung

void **pdf_save** (int PDF-Dokument)

Die Funktion **pdf_save()** sichert die aktuelle Umgebung. Sie wirkt wie das Postscript-Kommando gsave. Die Funktion ist sehr sinnvoll, wenn Sie einzelne Objekte drehen oder verschieben möchte ohne andere Objekte zu beienflussen. Auf **pdf_save()** sollte immer die Funktion pdf_restore() folgen, um die ürsprüngliche Umgebung wieder herzustellen.

Siehe auch pdf_restore().

pdf_restore (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Stellt eine zuvor gesicherte Umgebung wieder her

void **pdf_restore** (int PDF-Dokument)

Die Funktion **pdf_restore()** stellt eine frühere Umgebung wieder her, die mit pdf_save() gesichert wurde. Die Funktion arbeit wie das Postscript-Kommando grestore.

Beispiel 1. Save and Restore

```
<?php pdf_save($pdf);
// Alle möglichen Rotationen, Transformationen, ...
pdf_restore($pdf) ?>
```

Siehe auch pdf_save().

pdf_translate (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Ursprung des Koordinatenssystems

void **pdf_translate** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **pdf_translate()** setzt den Ursprung des Koordinatensystems auf den Punkt (*x-Koor*, *y-Koor*) relativ zum aktuellen Koordinatenursprung. Das folgende Beispiel zeichnet eine Linie vom Punkt (0, 0) zum Punkt (200, 200) relativ zum anfänglichen Ursprung. Die aktuelle Position muss nach **pdf_translate()** und vor dem weiteren Zeichnen gesetzt werden.

Beispiel 1. Translation

```
<?php pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
pdf_translate($pdf, 100, 100);
pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
?>
```

pdf_scale (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Skalierungsfaktor

void **pdf_scale** (int PDF-Dokument, double x-Skalierung, double y-Skalierung)

Die Funktion **pdf_scale()** setzt den Skalierungsfaktor in beiden Richtungen. Das folgende Beispiel skaliert in x- und y-Richtung um den Faktor 72. Die folgende Linie hat damit eine Länge von etwa 1.41 Inch.

Beispiel 1. Scaling

```
<?php pdf_scale($pdf, 72.0, 72.0);
pdf_lineto($pdf, 1, 1);
pdf_stroke($pdf);
?>
```

pdf_rotate (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Rotation

void **pdf_rotate** (int PDF-Dokument, double Winkel)

Die Funktion **pdf_rotate()** setzt die Drehung in auf *`Winkel`* Grad.

pdf_setflat (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets flatness

void **pdf_setflat** (int PDF-Dokument, double Wert)

Die Funktion **pdf_setflat()** setzt the flatness auf einen Wert zwischen 0 und 100.

pdf_setlinejoin (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Verbindungsart von Linien

void **pdf_setlinejoin** (int PDF-Dokument, long Wert)

Die Funktion **pdf_setlinejoin()** setzt die Verbindungsart von Linien auf einen Wert zwischen 0 und 2.

pdf_setlinecap (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Type der Linienenden

void **pdf_setlinecap** (int PDF-Dokument, int Typ)

Die Funktion **pdf_setlinecap()** setzt den Typ der Linienende auf einen Wert zwischen 0 und 2.

pdf_setmiterlimit (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets miter limit

void **pdf_setmiterlimit** (int PDF-Dokument, double Wert)

Die Funktion **pdf_setmiterlimit()** setzt the miter limit größer oder gleich 1.

pdf_setlinewidth (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Linienbreite

void **pdf_setlinewidth** (int PDF-Dokument, double Breite)

Die Funktion **pdf_setlinewidth()** setzt die Linienbreite auf *Breite* Punkte.

pdf_setdash (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt das Muster für gestrichelte Linien

void **pdf_setdash** (int PDF-Dokument, double Weiß, double Schwarz)

Die Funktion **pdf_setdash()** setzt das Muster für gestrichelte Linien auf *Weiß* weiße Punkte und *Schwarz* schwarze Punkte. Wenn beide Werte 0 sind wird die Linie durchgezogen.

pdf_moveto (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die aktuelle Position

void **pdf_moveto** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **pdf_moveto()** setzt die aktuelle Position auf den Punkt (*x-Koor*, *y-koor*).

pdf_curveto (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeichnet eine Kurve

void **pdf_curveto** (int PDF-Dokument, double x1, double y1, double x2, double y2, double x3, double y3)

Die Funktion **pdf_curveto()** zeichnet eine Bezier-Kurve vom aktuellen Punkt zu dem Punkt (*x3*, *y3*) unter Verwendung von (*x1*, *y1*) und (*x2*, *y2*) als Kontrollpunkte.

Siehe auch pdf_moveto(), pdf_lineto(), pdf_stroke().

pdf_lineto (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeichnet eine Linie

void **pdf_lineto** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **pdf_lineto()** zeichnet eine Linie vom aktuellen Punkt zu dem Punkt (*x-koor*, *y-koor*).

Siehe auch pdf_moveto(), pdf_curveto(), pdf_stroke().

pdf_circle (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeichnet einen Kreis

void **pdf_circle** (int PDF-Dokument, double x-Koor, double y-Koor, double Radius)

Die Funktion **pdf_circle()** zeichnet einen Kreis mit dem Mittelpunkt an der Position (*x-koor*, *y-koor*) und dem Radius *Radius*.

Siehe auch pdf_arc(), pdf_stroke().

pdf_arc (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeichnet einen Kreisbogen

void **pdf_arc** (int PDF-Dokument, double x-Koor, double y-Koor, double Radius, double Start, double Ende)

Die Funktion **pdf_arc()** zeichnet einen Kreisbogen mit dem Mittelpunkt an der Position (*x-koor*, *y-koor*) und dem Radius *radius*, angefangen beim Winkel *Start* und beendet mit dem Winkel *Ende*.

Siehe auch pdf_circle(), pdf_stroke().

pdf_rect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeichnet ein Rechteck

void **pdf_rect** (int PDF-Dokument, double x-koor, double y-koor, double Breite, double Höhe)

Die Funktion **pdf_rect()** zeichnet ein Rechteck mit der unteren Ecke an der Position (`x-Koor`, `y-Koor`). Die Breite wird auf `widgth` gesetzt. Die Höhe wird auf `height` gesetzt.

Siehe auch pdf_stroke().

pdf_closepath (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schließt einen Pfad

void **pdf_closepath** (int PDF-Dokument)

Die Funktion **pdf_closepath()** schließt den aktuellen Pfad. Viele Funktion wie z.B. pdf_moveto(), pdf_circle() und pdf_rect() starten einen neuen Pfad.

pdf_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeichnet eine Linie entlang eines Pfades

void **pdf_stroke** (int PDF-Dokument)

Die Funktion **pdf_stroke()** zeichnet eine Linie entlang des aktuellen Pfads. Der aktuelle Pfad setzt sich beispielsweise aus allen Anweisung, die eine Linie gezeichnet haben zusammen. Ohne Aufruf dieser Funktion würde die Linie nicht erscheinen.

Siehe auch pdf_closepath(), pdf_closepath_stroke().

pdf_closepath_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schließt einen Pfad und zeichnet eine Linie entlang des Pfades

void **pdf_closepath_stroke** (int PDF-Dokument)

Die Funktion **pdf_closepath_stroke()** ist eine Kombination von pdf_closepath() und pdf_stroke(). Sie löscht gleichzeitig den Pfad.

Siehe auch pdf_closepath(), pdf_stroke().

pdf_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Füllt den aktuellen Pfad

void **pdf_fill** (int PDF-Dokument)

Die Funktion **pdf_fill()** füllt das Innere des aktuellen Pfads mit der aktuellen Füllfarbe.

Siehe auch pdf_closepath(), pdf_stroke(), pdf_setgray_fill(), pdf_setgray(), pdf_setrgbcolor_fill(), pdf_setrgbcolor().

pdf_fill_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Füllt den aktuellen Pfad und zeichnet eine Linie entlang des Pfades

void **pdf_fill_stroke** (int PDF-Dokument)

Die Funktion **pdf_fill_stroke()** füllt das Innere des aktuellen Pfads mit der aktuellen Füllfarbe und zeichnet zudem eine Linie entlang des Pfads.

Siehe auch pdf_closepath(), pdf_stroke(), pdf_fill(), pdf_setgray_fill(), pdf_setgray(), pdf_setrgbcolor_fill(), pdf_setrgbcolor().

pdf_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Schließt, füllt und zeichnet eine Linie entlang des Pfades

void **pdf_closepath_fill_stroke** (int PDF-Dokument)

Die Funktion **pdf_closepath_fill_stroke()** schließt, füllt das Innere des aktuellen Pfads mit der aktuellen Füllfarbe and zeichnet ein Linie entlang des Pfads.

Siehe auch pdf_closepath(), pdf_stroke(), pdf_fill(), pdf_setgray_fill(), pdf_setgray(), pdf_setrgbcolor_fill(), pdf_setrgbcolor().

pdf_endpath (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Beendet den aktuellen Pfad

void **pdf_endpath** (int PDF-Dokument)

Die Funktion **pdf_endpath()** beendet den aktuellen Pfad ohne ihn zu schließen.

Siehe auch pdf_closepath().

pdf_clip (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Begrenzt alle Zeichenoperation auf den aktuellen Pfad

void **pdf_clip** (int PDF-Dokument)

Die Funktion **pdf_clip()** begrenzt alle Zeichenoperationen auf das innere des aktuellen Pfads.

pdf_setgray_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Füllfarbe auf einen Grauwert

void **pdf_setgray_fill** (int PDF-Dokument, double Grauwert)

Die Funktion **pdf_setgray_fill()** setzt den aktuellen Grauwert mit dem ein Pfad gefüllt wird.

Siehe auch pdf_setrgbcolor_fill().

pdf_setgray_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Zeichenfarbe auf einen Grauwert

void **pdf_setgray_stroke** (int PDF-Dokument, double Grauwert)

Die Funktion **pdf_setgray_stroke()** setzt die aktuellen Grauwert zum Zeichen von Linien.

Siehe auch pdf_setrgbcolor_stroke().

pdf_setgray (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Zeichen- und Füllfarbe auf einen Grauwert

void **pdf_setgray** (int PDF-Dokument, double Grauwert)

Die Funktion **pdf_setgray()** setzt den aktuellen Grauwert zum Zeichen und Füllen.

Siehe auch pdf_setrgbcolor_stroke(), pdf_setrgbcolor_fill().

pdf_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Füllfarbe auf einen Farbwert

void **pdf_setrgbcolor_fill** (int PDF-Dokument, double Rotwert, double Grünwert, double Blauwert)

Die Funktion **pdf_setrgbcolor_fill()** setzt die aktuelle Füllfarbe auf einen RGB-Farbwert.

Siehe auch **pdf_setrgbcolor_fill()**.

pdf_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Zeichenfarbe auf einen Farbwert

void **pdf_setrgbcolor_stroke** (int PDF-Dokument, double Rotwert, double Grünwert, double Blauwert)

Die Funktion **pdf_setrgbcolor_stroke()** setzt die aktuelle Zeichenfarbe auf einen RGB-Farbwert.

Siehe auch **pdf_setrgbcolor_stroke()**.

pdf_setrgbcolor (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Zeichen- und Füllfarbe auf einen Farbwert

void **pdf_setrgbcolor** (int PDF-Dokument, double Rotwert, double Grünwert, double Blauwert)

Die Funktion pdf_setrgbcolor_stroke() setzt sowohl die aktuelle Füll- und Zeichenfarbe auf einen RGB-Farbwert.

Siehe auch pdf_setrgbcolor_stroke(), pdf_setrgbcolor_fill().

pdf_add_outline (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fügt Lesemarke zur aktuellen Seite hinzu

int **pdf_add_outline** (int PDF-Dokument, string Text [, int Mutter [, int offen]])

Die Funktion **pdf_add_outline()** fügt der aktuellen Seite eine Lesemarke mit dem Text *text* hinzu.

Die Lesemarke wird als Kind von *Mutter* hinzugefügt und wird geöffnet dargestellt von der Parameter *offen* ungleich 0 ist. Der Rückgabewert ist die Kennung der Lesemarke und kann als Mutter weiterer Lesemarken dienen. Damit lassen sich Hierarchien von Lesemarken aufbauen.

Leider macht pdflib keine Kopie des Textes, so dass dies in PHP geschehen muss. Zur Zeit wird dieser Text nicht wieder freigegeben, allerdings kümmert sich der PHP-Speichermanager darum.

pdf_set_transition (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt den Übergang zur nächsten Seite

void **pdf_set_transition** (int PDF-Dokument, int Transition)

Die Funktion **pdf_set_transition()** setzt die Art der Transition zur folgenden Seite. Die möglichen Werte des Parameters *transition* sind:
0 für keine,
1 für zwei, über den Bildschirm kriechende Linien, die die neue Seite entstehen lassen,
2 für mehrere, über den Bildschirm kriechende Linien, die die neue Seite entstehen lassen,
3 für einen Kasten der die Seite entstehen läßt,
4 für eine einzelne, über den Bildschirm kriechende Linie, die die neue Seite entstehen lassen,
5, wenn die Seite sich auflösen soll,
6 für eine Auflösungseffekt, der sich über die Seite bewegt,
7, wenn die Seite von der neuen Seite einfach ersetzt wird (voreingestellt).

Siehe auch pdf_set_duration().

pdf_set_duration (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt die Zeitdauer bis zur nächsten Seite

void **pdf_set_duration** (int PDF-Dokument, double Dauer)

Die Funktion **pdf_set_duration()** setzt die Dauer bis zur nächsten Seite automatisch umgeblättert wird.

Siehe auch pdf_set_transition().

pdf_open_gif (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Öffnet ein GIF-Bild

int **pdf_open_gif** (int PDF-Dokument, string Dateiname)

Die Funktion **pdf_open_gif()** öffnet ein Bild im GIF-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

> **Anmerkung:** Diese Funktion sollte nicht mehr verwendet werden. Benutzen Sie stattdessen die Funktion pdf_open_image_file().

Beispiel 1. Einbinden eines GIF Bildes

```
<?php
$im = pdf_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

Siehe auch pdf_close_image(), pdf_open_jpeg(), pdf_open_png(), pdf_open_tiff(), pdf_open_memory_image(), pdf_execute_image(), pdf_place_image(), pdf_put_image().

pdf_open_png (PHP 4 >= 4.0.0)

Öffnet ein PNG-Bild

int **pdf_open_png** (int PDF-Dokument, string Dateiname)

Die Funktion **pdf_open_png()** öffnet ein Bild im PNG-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

> **Anmerkung:** Diese Funktion sollte nicht mehr verwendet werden. Benutzen Sie stattdessen die Funktion pdf_open_image_file().

Beispiel 1. Einbinden eines PNG Bildes

```
<?php
$im = pdf_open_png($pdf, "test.png");
pdf_place_image($pdf, $im, 100, 100, 1);
```

```
pdf_close_image($pdf, $im);
?>
```

Siehe auch pdf_close_image(), pdf_open_jpeg(), pdf_open_gif(), pdf_open_tiff(), pdf_open_memory_image(), pdf_execute_image(), pdf_place_image(), pdf_put_image().

pdf_open_image_file (PHP 3 CVS only, PHP 4 >= 4.0.0)

Liest ein Bild aus einer Datei

int **pdf_open_image_file** (int PDF-Dokument, string Bildformat, string Dateiname)

Die Funktion **pdf_open_image_file()** liest ein Bild des Formats *Bildformat* aus der Datei *Dateiname* und macht es für das PDF-Dokument verfügbar. Als Bildformat sind 'png', 'tiff', 'jpeg' und 'gif' erlaubt. Die Funktion liefert ein PDF-Bild zurück.

Beispiel 1. Einfügen eines Bildes

```
<?php
$pim = pdf_open_image_file($pdf, "png", "picture.png");
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

Siehe auch pdf_close_image(), pdf_open_jpeg(), pdf_open_gif(), pdf_open_tiff(), pdf_open_png(), pdf_execute_image(), pdf_place_image(), pdf_put_image().

pdf_open_memory_image (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liest ein Bild, das mit PHP erzeugt wurde

int **pdf_open_memory_image** (int PDF-Dokument, int Bild)

Die Funktion **pdf_open_memory_image()** nimmt ein Bild, das mit den PHP eigenen Funktion zur Bilderzeugung erstellt wurde und macht es für das PDF-Dokument verfügbar. Die Funktion liefert ein PDF-Bild zurück.

Beispiel 1. Einfügen eines Bildes

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
```

```
?>
```

pdf_open_jpeg (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Öffnet ein JPEG-Bild

int **pdf_open_jpeg** (int PDF-Dokument, string Dateiname)

Die Funktion **pdf_open_jpeg()** öffnet ein Bild im JPEG-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

Siehe auch pdf_close_image(), pdf_open_gif(), pdf_open_tiff(), pdf_open_png(), pdf_open_memory_image(), pdf_execute_image(), pdf_place_image(), pdf_put_image().

pdf_open_tiff (PHP 4 >= 4.0.0)

Öffnet ein TIFF-Bild

int **pdf_open_tiff** (int PDF-Dokument, string Dateiname)

Die Funktion **pdf_open_tiff()** öffnet ein Bild im TIFF-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

Anmerkung: Diese Funktion sollte nicht mehr verwendet werden. Benutzen Sie stattdessen die Funktion pdf_open_image_file().

Siehe auch pdf_close_image(), pdf_open_gif(), pdf_open_jpeg(), pdf_open_png(), pdf_open_memory_image(), pdf_execute_image(), pdf_place_image(), pdf_put_image().

pdf_close_image (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Schließt ein Bild

void **pdf_close_image** (int Bild)

Die Funktion **pdf_close_image()** schließt ein Bild, das zuvor mit einer der **pdf_open_xxx()** Funktionen geöffnet wurde.

Siehe auch pdf_open_jpeg(), pdf_open_gif(), pdf_open_png(), pdf_open_tiff(), pdf_open_memory_image().

pdf_place_image (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Plaziert ein Bild auf der Seite

void **pdf_place_image** (int PDF-Dokument, int Bild, double x-Koor, double y-Koor, double scale)

Die Funktion **pdf_place_image()** plaziert ein Bild auf der Seite an Position (*x-Koor*, *x-Koor*). Das Bild kann gleichzeitig skaliert werden.

Siehe auch pdf_put_image().

pdf_put_image (PHP 3>= 3.0.7)

Speichert ein Bild im PDF-Dokument für späteren Gebrauch

void **pdf_put_image** (int PDF-Dokument, int image)

Die Funktion **pdf_put_image()** plaziert ein Bild in dem PDF-Dokument ohne es anzuzeigen. Das gespeichert Bild can durch die Funktion pdf_execute_image() so oft wie notwendig angezeigt werden. Dies ist sinnvoll, wenn das gleiche Bild mehrfach genutzt werden soll, um die Dateigröße des PDF-Dokuments klein zu halten. Der Gebrauch von **pdf_put_image()** und pdf_execute_image() ist insbesonder bei größeren Bilder zu empfehlen, wenn sie mehrfach verwerdet werden.

Anmerkung: Diese Funktion ist bedeutungslos seit der Version 2.01 von pdflib. Es wird lediglich eine Warnung ausgegeben.

Siehe auch **pdf_put_image()**, pdf_place_image(), pdf_execute_image().

pdf_execute_image (PHP 3>= 3.0.7)

Plaziert ein gespeichertes Bild auf der Seite

void **pdf_execute_image** (int PDF-Dokument, int image, double x-Koor, double y-Koor, double Skalierung)

Die Funktion **pdf_execute_image()** zeigt ein Bild, das zuvor mit pdf_put_image() in dem PDF-Dokument abgelegt wurde.

Das Bild kann gleichzeitig skaliert werden. Ein Skalierungsfaktor von 1.0 zeigt das Bild in Originalgröße.

Anmerkung: Diese Funktion ist bedeutungslos seit der Version 2.01 von pdflib. Es wird lediglich eine Warnung ausgegeben.

Beispiel 1. Multiple show of an image

```
<?php
$im = ImageCreate(100, 100);
$col1 = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col1);
$pim = pdf_open_memory_image($pdf, $im);
pdf_put_image($pdf, $pim);
pdf_execute_image($pdf, $pim, 100, 100, 1);
pdf_execute_image($pdf, $pim, 200, 200, 2);
pdf_close_image($pdf, $pim);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Fügt eine Anmerkung hinzu

void **pdf_add_annotation** (int PDF-Dokument, double llx, double lly, double urx, double ury, string Titel, string Inhalt)

Die Funktion **pdf_add_annotation()** fügt eine Anmerkung mit der unteren linken Ecke an der Position (*llx*, *lly*) und der oberen rechten Ecke an der Position (*urx*, *ury*) ein.

pdf_set_border_style (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Setzt den Stil der Umrandung von Verweisen und Anmerkungen

void **pdf_set_border_style** (int PDF-Dokument, string Stil, double Breite)

Die Funktion **pdf_set_border_style()** setzt den Stil und die Breite der Umrandung von Verweisen und Anmerkungen. Der Parameter *Stil* kann die Werte 'solid' oder 'dashed' annehmen.

Siehe auch pdf_set_border_color(), pdf_set_border_dash().

pdf_set_border_color (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Setzt die Farbe der Umrandung von Verweisen und Anmerkungen

void **pdf_set_border_color** (int PDF-Dokument, double Rot, double Grün, double Blua)

Die Funktion **pdf_set_border_color()** setzt die Farbe der Umrandung von Verweisen und Anmerkungen. Die drei Farbkomponenten haben eine Wert zwischen 0.0 und 1.0.

Siehe auch pdf_set_border_style(), pdf_set_border_dash().

pdf_set_border_dash (PHP 4)

Setzt das Muster für gestrichelte Linien als Umrandung von Verweisen und Anmerkungen

void **pdf_set_border_dash** (int PDF-Dokument, double Schwarz, double Weiß)

Die Funktion **pdf_set_border_dash()** setzt die Länge der schwarzen und weißen Bereiche einer gestrichelten Umrandung von Verweisen und Anmerkungen. links and annotations.

Siehe auch pdf_set_border_style(), pdf_set_border_color().

LXXV. Verisign Payflow Pro Funktionen

Diese Erweiterung macht es möglich, Geschäftsabwicklungen via Kreditkarte und andere Finanztransaktionen über die Verisign Payment Services, auch als Signio bekannt, abzuwickeln (http://www.verisign.com/payment/).

Diese Funktionen sind nur verfügbar, wenn PHP mit der Option `--with-pfpro[=DIR]` compiliert wurde. Dazu ist das SDK nötig, dass im Management-Interface (https://testmanager.signio.com/Downloads/Downloads_secure.htm) heruntergeladen werden kann, sobald man sich registriert hat. Sollte diese Erweiterung in einem SSL-Webserver oder zusammen mit anderen SSL-Komponenten verwendet werden, muss die Beta-Version des SDK verwendet werden.

Sobald das SDK heruntergeladen wurde, sollte man die Dateien aus dem `lib`-Verzeichnis der Distribution kopieren. Die Header-Datei `pfpro.h` sollte in das Verzeichnis `/usr/local/include`, die Bibliothek `libpfpro.so` in das Verzeichnis `/usr/local/lib` kopiert werden.

Wenn diese Funktionen genutzt werden, ist es mögliche, auf Aufrufe von pfpro_init() und pfpro_cleanup() zu verzichten, da diese automatisch aufgerufen werden, wenn sie benötigt werden. Diese Funktionen sind jedoch nachwievor für Sonderfälle vorhanden, z.B. wenn viele Transaktionen auf einmal ausgeführt werden sollen, oder genaue Kontrolle über die Biblitohek benötigt wird. Mehrere Transaktionen zwischen diesen beiden Funktionen können mittels der pfpro_process()-Funktion ausgeführt werden.

Diese Funktionen sind seit PHP 4.0.2 verfügbar.

> **Anmerkung:** Diese Funktionen stellen nur eine Schnittstelle zu den Verisign Payment Services dar. Details über die nötigen Paramter sind in dem Dokument Payflow Pro Developers Guide zu finden, welches bei Verisign erhältlich ist.

pfpro_init (PHP 4 >= 4.0.2)

Initialisiert die Payflow Pro Bibliothek

void **pfpro_init** (void)

pfpro_init() wird benutzt, um die Payflow Pro Bibliothek zu initialisieren. Auf einen Aufruf dieser Funktion kann verzichtet werden, da **pfpro_init()** automatisch vor der ersten Transaktion aufgerufen wird.

See also pfpro_cleanup().

pfpro_cleanup (PHP 4 >= 4.0.2)

Schliesst Payflow Pro Bibliothek

void **pfpro_cleanup** (void)

pfpro_cleanup() veranlasst eine saubere Schliessung der Payflow Pro-Bibliothek. Auf einen Aufruf dieser Funktion kann verzichtet werden, da die Erweiterung automatisch ein **pfpro_cleanup()** ausführen wird, nachdem das Script beendet wurde.

Siehe auch pfpro_init().

pfpro_process (PHP 4 >= 4.0.2)

Führt eine Transaktion mit Payflow Pro aus

array **pfpro_process** (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])

Gibt einen assoziativen Array, der die Antwort enthält, zurück.

pfpro_process() führt eine Transaktion mir Payflow Pro aus. Der erste Parameter ist ein assoziativer Array, der die Schlüssel und Werte enthält. Diese werden kodiert und an den PFPro-Prozessor weitergegeben.

Der zweite Parameter ist optional und gibt den Host an, zu dem das Script sich verbinden soll. Voreingestellt ist "test.signio.com", normalerweise sollte dies in "connect.signio.com" geändert werden, um Transaktionen durchzuführen.

Der dritte Parameter gibt den Port an, auf den das Script sicher verbinden soll. Voreingstellt ist 443, der Standard-SSL-Port.

Der vierte Parameter gibt den Timeout-Wert in Sekunden an. Voreingestellt ist der Wert 30 Sekunden. Dieser Timeout gilt jedoch nur zu Beginn des Scripts, also kann es durchaus sehr lange dauern, wenn ein Netzwerkproblem auftritt, nachdem die Verbindung erfolgreich hergestellt wurde.

Der fünfte Parameter gibt den SSL-Proxy an, falls dieser benötigt wird. Der sechste Parameter gibt den Port des Proxys an.

Der siebte und der achte Parameter geben den Login-Namen und das Login-Passwort an.

Die Funktion gibt einen assoziativen Array der Schlüssel und Werte zurück.

Anmerkung: Bitte lesen sie die Payflow Pro Entwicklerdokumentation für alle Details zu den erforderlichen Parametern.

Beispiel 1. Payflow Pro Beispiel

```
<?php

pfpro_init();

$transaction = array(USER => 'mylogin',
     PWD => 'mypassword',
     TRXTYPE => 'S',
     TENDER => 'C',
     AMT => 1.50,
     ACCT => '4111111111111111',
     EXPDATE => '0904'
     );

$response = pfpro_process($transaction);

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign response code was ".$response[RESULT];
echo ", which means: ".$response[RESPMSG]."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_process_raw (PHP 4 >= 4.0.2)

Führt eine rohe Transaktion mit Payflow Pro aus

string **pfpro_process_raw** (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])

Gibt die Antwort in einem String zurück.

pfpro_process_raw() führt eine rohe Transaktion mit Payflow Pro durch. Es sollte jedoch auf jeden Fall pfpro_process() verwendet werden, da sonst keinerlei Normung vorliegt.

Der erste Parameter ist ein String, der die rohe Transaktionsanfrage enthält. Alle anderen Parameter sind die Selben wie bei pfpro_process(). Der Rückgabewert ist ein String, der die rohe Antwort enthält.

Anmerkung: Bitte lesen sie die Payflow Pro Entwicklerdokumentation für alle Details zu den erforderlichen Parametern. Es wird empfohlen, die Funktion pfpro_process() zu verwenden.

Beispiel 1. Payflow Pro Beispiel für den Raw-Modus

```
<?php

pfpro_init();

$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=411

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pfpro_cleanup();

?>
```

pfpro_version (PHP 4 >= 4.0.2)

Gibt die Versionsinformationen der Payflow Pro Software zurück

string **pfpro_version** (void)

pfpro_version() gibt die Versionsinformationen der Payflow Pro Bibliothek zurück.

LXXVI. PHP Optionen und Informationen

assert (PHP 4 >= 4.0.0)

Prüft ab, ob eine Bedingung oder Abfrage `FALSE` ist

int **assert** (string|bool assertion)

assert() überprüft den übergebenen Parameter *`assertion`* und führt ein über die Funktion assert_options() definiertes Ereignis aus, falls die in *`assertion`* spezifizierte Bedingung `FALSE` ist.

Wird der Parameter *`assertion`* als String übergeben, so wird der String als PHP-Code interpretiert. Der Vorteil der Übergabe des Parameters *`assertion`* als Zeichenkette liegt darin, dass die ausgegebene Meldung die über *`assertion`* übergebene Zeichenkette enthält.

assert() sollte nur zum Debuggen des Codes und nicht in Produktionsumgebungen verwendet werden werden, beispielsweise zur Überprüfung von Eingaben.

Die Optionen für die **assert()** können über die Funktion assert_options() oder in den Einstellungen der .ini-Datei gesetzt werden.

Die Funktion assert_options() mit der ASSERT_CALLBACK Direktive ermöglicht eine Funktion anzugeben, die bei fehlender Überprüfung aufgerufen wird.

Callback-Funktionen in Zusammenhang mit **assert()** sind nützlich, um beispielsweise automatisierte Tests durchzuführen, da die Callback-Funktion pber die Information verfügt, von welcher Stelle sie aufgerufen wurde.

Die Callback-Funktion wird mit drei Parametern aufgerufen. Der erste Parameter enthält den Namen des Scripts, in dem die Überprüfung statt fand, der zweite Parameter die Zeilennummer, und der dritte Paramter enthält die über *`assertion`* angegebene Bedingung.

Beispiel 1. Überprüfung mit benutzerdefinierter Funktion

```
<?php
// Active assert and make it quiet
assert_options (ASSERT_ACTIVE, 1);
assert_options (ASSERT_WARNING, 0);
assert_options (ASSERT_QUIET_EVAL, 1);

// Create a handler function
function my_assert_handler ($file, $line, $code) {
    echo "<hr>Assertion Failed:
        File '$file'<br>
        Line '$line'<br>
        Code '$code'<br><hr>";
}

// Set up the callback
assert_options (ASSERT_CALLBACK, 'my_assert_handler');

// Make an assertion that should fail
assert ('mysql_query ("")');
?>
```

assert_options (PHP 4 >= 4.0.0)

Setzt oder liefert die Assert-Optionen

mixed **assert_options** (int what [, mixed value])

assert_options() ermöglicht es, die verschiedenene assert() Optionen zu setzen, oder abzufragen, welche Optionen gesetzt sind.

Tabelle 1. assert Optionen

Option	Parameter in .ini-Datei	Standardwert	Beschreibung
ASSERT_ACTIVE	assert.active	1	assert() Überprüfung aktivieren
ASSERT_WARNING	assert.warning	1	gibt eine PHP Warnung für jede fehlgeschlagene Überprüfung an
ASSERT_BAIL	assert.bail	0	Beendet das Programm bei fehlgeschlagener Überprüfung
ASSERT_QUIET_EVAL	assert.quiet_eval	0	Unterdrückt die Fehlerausgabe während der Überprüfung
ASSERT_CALLBACK	assert_callback	(`NULL`)	ermöglicht die Angabe einer benuterdefinierten Funktion

assert_options() gibt die Einstellung der gesetzten Optionen zurück oder liefert den Wert `FALSE` bei Auftreten eines Fehlers zurück.

dl (PHP 3, PHP 4 >= 4.0.0)

Eine PHP-Extension zur Laufzeit laden

int **dl** (string library)

Lädt die PHP-Bibliothek *`library`* zur Laufzeit. Siehe auch die Extension Loading-Direktiven.

extension_loaded (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Zeigt an, ob eine Bibliothek geladen wurde

bool **extension_loaded** (string name)

Ist wahr, wenn die Bibliothek, die mit *`name`* definiert wurde, geladen ist. Einen Überblick über die verschiedenen Bibliotheken erhält man mit phpinfo().

Siehe auch phpinfo().

Anmerkung: Diese Funktion wurde mit Version 3.0.10 eingeführt.

getenv (PHP 3, PHP 4 >= 4.0.0)

Zeigt den Wert einer Umgebungsvariablen an

string **getenv** (string varname)

Zeigt den Wert der Umgebungsvariablen *varname* an oder FALSE bei einem Fehler.

```
$ip = getenv ("REMOTE_ADDR"); // get the ip number of the user
```

Eine Liste aller Umgebungsvariablen wird mit phpinfo() angezeigt. Ein Überlick über viele Umgebungsvariablen und deren Bedeutung findet sich unter den CGI Spezifikationen (http://hoohoo.ncsa.uiuc.edu/cgi/), speziell auf der Seite der Umgebungsvariablen (http://hoohoo.ncsa.uiuc.edu/cgi/env.html).

get_cfg_var (PHP 3, PHP 4 >= 4.0.0)

Zeigt den Wert einer Option der PHP-Konfiguration

string **get_cfg_var** (string varname)

Zeigt den derzeitigen Wert der Variablen der PHP-Konfiguration, die mit *varname* ausgewählt wurde oder FALSE bei einem Fehler.

Bringt keine Informationen über die Konfiguration, mit der PHP kompilert wurde, oder die aus der Apache-Konfiguration gelesen wird (mittels der php3_configuration_option Anweisungen).

Um herauszubekommen, ob ein configuration file, benutzt wird, sollte versucht werden, den Wert der cfg_file_path Einstellung zu erhalten. Ist dieser Wert verfügbar, wird eine Konfigurationsdatei verwendet.

get_current_user (PHP 3, PHP 4 >= 4.0.0)

Den Besitzer des aktuellen PHP-Scripts anzeigen.

string **get_current_user** (void)

Zeigt den Namen des Besitzers des aktuellen PHP-Scripts an.

Siehe auch getmyuid(), getmypid(), getmyinode(), und getlastmod().

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeigt die aktuelle Konfiguration von magic quotes gpc

long **get_magic_quotes_gpc** (void)

Zeigt die aktuelle Konfiguration von magic_quotes_gpc. (0 für aus, 1 für an).

Siehe auch get_magic_quotes_runtime() und set_magic_quotes_runtime().

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zeigt die aktuelle Konfiguration von magic_quotes_runtime

Long **get_magic_quotes_runtime** (void)

Zeigt die aktuelle Konfiguration von magic_quotes_runtime. (0 für aus, 1 für an).

Siehe auch get_magic_quotes_gpc() und set_magic_quotes_runtime().

getlastmod (PHP 3, PHP 4 >= 4.0.0)

Zeigt die Uhrzeit der letzten Änderung einer Seite

int **getlastmod** (void)

Zeigt die Uhrzeit der letzten Änderung der aktuellen Seite. Der Wert wird als Unix-Timestamp angezeigt, welcher für date() benutzt werden kann. Bringt `FALSE` bei einem Fehler.

Beispiel 1. getlastmod() Beispiel

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: ".date ("F d Y H:i:s.", getlastmod());
```

Siehe auch date(), getmyuid(), get_current_user(), getmyinode() und getmypid().

getmyinode (PHP 3, PHP 4 >= 4.0.0)

Gibt den Inode des aktuellen Skripts aus

int **getmyinode** (void)

Zeigt den inode des aktuellen Scripts oder `FALSE` bei einem Fehler.

Siehe auch getmyuid(), get_current_user(), getmypid() und getlastmod().

getmypid (PHP 3, PHP 4 >= 4.0.0)

Zeigt die ID des PHP-Prozesses

int **getmypid** (void)

Zeigt die aktuelle ID des PHP-Prozesses oder FALSE bei einem Fehler.

Achtung: Wenn PHP als Server-Modul läuft, ist es nicht garantiert, dass Scripts unter verschiedenen PIDs laufen.

Siehe auch getmyuid(), get_current_user(), getmyinode() und getlastmod().

getmyuid (PHP 3, PHP 4 >= 4.0.0)

Zeigt die UID des Besitzers eines PHP-Scripts

int **getmyuid** (void)

Zeigt die User-ID des aktuellen Scripts oder FALSE bei einem Fehler.

Siehe auch getmypid(), get_current_user(), getmyinode() und getlastmod().

getrusage (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Zeigt den aktuellen Ressourcenverbrauch an

array **getrusage** ([int who])

Dies ist eine Schnittstlle zu getrusage(2). Stellt ein assoziatives Array mit den Daten zur Verfügung, die der Systemaufruf ausgibt. Wenn who 1 ist, wird getusage mit RUSAGE_CHILDREN aufgerufen.

Alle Einträge können über ihre Dukumentenfeldernamen aufgerufen werden.

Beispiel 1. getrusage() Beispiel

```
$dat = getrusage();
echo $dat["ru_nswap"];         # number of swaps
echo $dat["ru_majflt"];        # number of page faults
echo $dat["ru_utime.tv_sec"];  # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```

Siehe auch im Handbuch des Betriebssystems für weitere Details.

phpinfo (PHP 3, PHP 4 >= 4.0.0)

Zeigt viele Informationen zu PHP

int **phpinfo** (void)

Zeigt eine grosse Anzahl von Informationen über die aktuelle Konfiguration von PHP an. Unter anderem die Optionen während der Kompilierens und die Erweiterungen, die PHP Version, Informationen über den Server, die Umgebung (wenn PHP als Modul kompiliert wurde), die PHP Umgebung, Version und Informationen zum Betriebssystem, Pfade, Haupt- and lokale Werte der Konfigurationsoptionen, HTTP Header, und die GNU Public License.

Siehe auch phpversion().

php_logo_guid (PHP 4 >= 4.0.0)

Die logo guid erhalten

string **php_logo_guid** (void)

Anmerkung: Diese Funktion wurde mit Version PHP4 Beta 4 eingeführt.

php_sapi_name (PHP 4)

Zeigt die Schnittstelle zwischen Webserver und PHP an

string **php_sapi_name** (void)

php_sapi_name() zeigt die Schnittstelle zwischen Webserver und PHP (Server API, SAPI) als kleingeschriebene Zeichenkette an. Bei CGI PHP ist diese Zeichenkette "cgi", bei mod_php für Apache ist die Zeichenkette "apache" usw.

Beispiel 1. php_sapi_name() Beispiel

```
$inter_type = php_sapi_name();
if ($inter_type == "cgi")
    print "Sie benutzen CGI PHP\n";
else
    print "Sie benutzen nicht CGI PHP\n";
```

phpversion (PHP 3, PHP 4 >= 4.0.0)

Zeigt die aktuell installierte PHP-Version

string **phpversion** (void)

Zeigt die Version des installierten PHP-Parsers.

Beispiel 1. phpversion() Beispiel

```
// Gibt z.B. 'Aktuelle PHP Version: 3.0rel-dev' aus
echo "Aktuelle PHP Version: ".phpversion();
```

Siehe auch phpinfo().

putenv (PHP 3, PHP 4 >= 4.0.0)

Setzt den Wert einer Umgebungsvariablen.

void **putenv** (string setting)

Setzt *setting* als Umgebungsvariable.

Beispiel 1. Setzen einer Umgebungsvariablen

```
putenv ("UNIQID=$uniqid");
```

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt magic_quotes_runtime

long **set_magic_quotes_runtime** (int new_setting)

Setzt magic_quotes_runtime. (0 für aus, 1 für an).

Siehe auch get_magic_quotes_gpc() und get_magic_quotes_runtime().

set_time_limit (PHP 3, PHP 4 >= 4.0.0)

Setzt die maximale Ausführungszeit

void **set_time_limit** (int seconds)

Legt die Zeit in Sekunden fest, die ein Script laufen darf. Ist diese Zeit abgelaufen, wird ein Fehler zurückgegeben. Der Standard liegt bei 30 Sekunden, es sei denn, max_execution_time ist configuration file eingetragen. Bei einer Einstellung von 0 Sekunden ist die maximale Ausführungszeit nicht begrenzt.

Wenn **set_time_limit()** aufgerufen wird, dann startet der Zähler neu. Das heisst, wenn die Standardeinstellung 30 Sekunden beträgt und nach 25 Sekunden durch ein Script ein Aufruf wie z.B. `set_time_limit(20)` erfolgt, darf das Script insgesamt 45 Sekunden laufen, bevor eine Fehlermeldung ausgegeben wird.

Achtung: Wenn PHP im Safemode läuft, hat **set_time_limit()** keinen Effekt. Die einzige Möglichkeit, dies zu umgehen, ist den Safemode im configuration file auszuschalten.

zend_logo_guid (PHP 4 >= 4.0.0)

Zeigt die Zend guid

string **zend_logo_guid** (void)

Anmerkung: Diese Funktion wurde mit Version PHP4 Beta 4 eingeführt.

get_loaded_extensions (PHP 4 >= 4.0.0)

Stellt ein Array mit den Namen aller Module, die kompiliert und geladen sind, zur Verfügung

array **get_loaded_extensions** (void)

Dieser Funktion zeigt die Namen aller Module an, die in den PHP-Interpreter kompiliert und geladen wurden

Zum Beispiel gibt diese Zeile

```
print_r (get_loaded_extensions());
```

eine Liste wie die folgende aus:

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
    [3] => session
    [4] => posix
    [5] => pgsql
    [6] => pcre
    [7] => gd
    [8] => ftp
    [9] => db
    [10] => Calendar
    [11] => bcmath
)
```

Siehe auch: get_extension_funcs()

get_extension_funcs (PHP 4 >= 4.0.0)

Stellt ein Array mit den Namen einer Funktion eines Moduls zur Verfügung

array **get_extension_funcs** (string module_name)

Diese Funktion zeigt die Namen aller Funktionen, die im Modul, welches von *module_name* angezeigt wird, an.

Zum Beispiel würden diese Zeilen

```
print_r(get_extension_funcs("xml"));
print_r(get_extension_funcs("gd"));
```

eine Liste von Funktionen in den Modulen `xml` bzw. `gd` ausgeben.

Siehe auch: get_loaded_extensions()

get_required_files (PHP 4 >= 4.0.0)

Stellt ein Array mit den Namen der Dateien zur Verfügung, die über require_once() in einem Script aufgerufen wurden

array **get_required_files** (void)

Diese Funktion stellt ein assoziatives Array der Namen aller Dateien zur Verfügung, die mit require_once() in ein Script geladen wurden. Die Indexe dieses Arrays sind die Dateinamen, die in require_once() benutzt werden, ohne die Endung ".php".

Dieses Beispiel

Beispiel 1. Ausgabe der required und included Dateien

```
<?php

require_once("local.php");
require_once("../inc/global.php");

for ($i=1; $i<5; $i++)
include "util".$i."php";

echo "Required_once files\n";
print_r (get_required_files());

echo "Included_once files\n";
print_r (get_included_files());
```

wird folgendes ausgeben:

```
Required_once files
Array
(
```

```
    [local] => local.php
    [../inc/global] => /full/path/to/inc/global.php
)

Included_once files
Array
(
    [util1] => util1.php
    [util2] => util2.php
    [util3] => util3.php
    [util4] => util4.php
)
```

Anmerkung: Ab PHP 4.0.1pl2 setzt diese Funktion vorraus, daß die Dateien, die mit `required_once` aufgerufen wurden, die Endung ".php" haben. Andere Endungen funktionieren nicht.

Siehe auch: require_once(), include_once() und get_included_files().

get_included_files (PHP 4 >= 4.0.0)

Stellt ein Array der Dateinamen zur Verfügung, die mittels include_once() in ein Script geladen wurden

array **get_included_files** (void)

Diese Funktion stellt ein assoziatives Array der Namen aller Dateien zur Verfügung, die mit include_once() in ein Script geladen wurden. Die Indexe dieses Arrays sind die Dateinamen, die in include_once() benutzt wurden, ohne die Endung ".php".

Anmerkung: Ab PHP 4.0.1pl2 setzt diese Funktion vorraus, daß die Dateien, die mit `include_once`aufgerufen wurden, die Endung ".php" haben. Andere Endungen funktionieren nicht.

Siehe auch: require_once(), include_once() und get_required_files().

LXXVII. POSIX Funktionen

Dieses Modul enthält eine Schnittstelle zu den im IEEE 1003.1 (POSIX.1) Standards Dokument definierten Funktionen, die durch andere Mittel nicht erreichbar sind. POSIX.1 definierte zum Beispiel auch die open(), read(), write() und close() Funktionen, die seit langer Zeit Bestandteil von PHP 3 sind. Trotzdem standen einige andere systemspezifische Funktionen bisher nicht zur Verfügung. Dieses Modul versucht hier Abhilfe zu schaffen, indem es einen einfachen Zugang zu diesen Funktionen bereitstellt.

posix_kill (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sendet einem Prozess ein Signal

bool **posix_kill** (int pid, int sig)

Diese Funktion sendet das Signal *sig* zum Prozess mit der Prozesskennung (Prozess-ID) *pid*. Sie gibt FALSE zurück, wenn sie das Signal nicht senden konnte, andernfalls TRUE.

Sehen Sie sich dazu auch die kill(2)-Manpage Ihres POSIX-Systems an, die zusätzliche Informationen über negative Prozess-IDs, die speziellen pid 0 und -1 und die Signalnummer 0 enthält.

posix_getpid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die aktuelle Prozesskennung

int **posix_getpid** (void)

Diese Funktion gibt die Prozesskennung des aktuellen Prozesses zurück.

posix_getppid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die Prozesskennung des Elternprozesses

int **posix_getppid** (void)

Diese Funktion gibt die Prozesskennung des Elternprozesses des aktuellen Prozesses zurück.

posix_getuid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die reale Benutzer-ID des aktuellen Prozesses

int **posix_getuid** (void)

Diese Funktion gibt die reale numerische Benutzer-ID des aktuellen Prozesses zurück.

Für Informationen darüber, wie die Benutzer-ID in einen gebräuchlichen Benutzernamen umgewandelt werden kann, siehe auch posix_getpwuid().

posix_geteuid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die effektive Benutzer-ID des aktuellen Prozesses

int **posix_geteuid** (void)

Diese Funktion gibt die effektive numerische Benutzer-ID des aktuellen Prozesses zurück.

Für Informationen darüber, wie die Benutzer-ID in einen gebräuchlichen Benutzernamen umgewandelt werden kann, siehe auch posix_getpwuid().

posix_getgid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die reale Gruppen-ID des aktuellen Prozesses

int **posix_getgid** (void)

Diese Funktion gibt die reale numerische Gruppen-ID des aktuellen Prozesses zurück.

Für Informationen darüber, wie die Gruppen-ID in einen gebräuchlichen Gruppennamen umgewandelt werden kann, siehe auch posix_getgrgid().

posix_getegid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die effektive Gruppen-ID des aktuellen Prozesses

int **posix_getegid** (void)

Diese Funktion gibt die effektive numerische Gruppen-ID des aktuellen Prozesses zurück.

Für Informationen darüber, wie die Gruppen-ID in einen gebräuchlichen Gruppennamen umgewandelt werden kann, siehe auch posix_getgrgid().

posix_setuid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Setzt die Benutzer-ID des aktuellen Prozesses

bool **posix_setuid** (int uid)

Diese Funktion setzt die reale Benutzer-ID *uid* des aktuellen Prozesses. Es handelt sich um eine privilegierte Funktion, für die Sie die entspechenden Berechtigungen (normalerweise root) auf Ihrem System benötigen, um diese Funktion ausführen zu können.

Gibt bei Erfolg TRUE und andernfalls FALSE zurück.

Siehe auch: posix_setgid().

posix_seteuid (PHP 4 >= 4.0.2)

Setzt die effektive Benutzer-ID des aktuellen Prozesses

bool **posix_seteuid** (int uid)

Diese Funktion setzt die effektive Benutzer-ID *uid* des aktuellen Prozesses. Es handelt sich um eine privilegierte Funktion, für die Sie die entspechenden Berechtigungen (normalerweise root) auf Ihrem System benötigen, um diese Funktion ausführen zu können.

Gibt bei Erfolg TRUE und andernfalls FALSE zurück.

Siehe auch: posix_setgid().

posix_setgid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Setzt die Gruppen-ID des aktuellen Prozesses

bool **posix_setgid** (int gid)

Diese Funktion setzt die reale Gruppen-ID *gid* des aktuellen Prozesses. Es handelt sich um eine privilegierte Funktion, für die Sie die entspechenden Berechtigungen (normalerweise root) auf Ihrem System benötigen, um diese Funktion ausführen zu können. Die sachgemäße Reihenfolge der Funktionsaufrufe ist zuerst **posix_setgid()** und zuletzt posix_setuid().

Gibt bei Erfolg TRUE und andernfalls FALSE zurück.

posix_setegid (PHP 4 >= 4.0.2)

Setzt die effektive Gruppen-ID des aktuellen Prozesses

bool **posix_setegid** (int gid)

Diese Funktion setzt die effektive Gruppen-ID des aktuellen Prozesses. Es handelt sich um eine privilegierte Funktion, für die Sie die entspechenden Berechtigungen (normalerweise root) auf Ihrem System benötigen, um diese Funktion ausführen zu können.

Gibt bei Erfolg TRUE und andernfalls FALSE zurück.

posix_getgroups (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die Gruppenliste des aktuellen Prozesses

array **posix_getgroups** (void)

Diese Funktion gibt ein Array von ganzen Zahlen zurück, das die numerischen Gruppen-IDs der Gruppenliste des aktuellen Prozesses enthält.

Für Informationen darüber, wie die Gruppen-IDs in gebräuchliche Gruppennamen umgewandelt werden können, siehe auch posix_getgrgid().

posix_getlogin (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert den Benutzernamen des Eigentümers des aktuellen Prozesses

string **posix_getlogin** (void)

Diese Funktion gibt den Benutzernamen des Benutzers zurück, dem der aktuelle Prozess gehört.

Für Informationen darüber, wie mehr Informationen über diesen Benutzer bekommen werden können, siehe posix_getpwnam().

posix_getpgrp (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die Prozessgruppenkennung des aktuellen Prozesses

int **posix_getpgrp** (void)

Diese Funktion gibt die Prozessgruppenkennung des aktuellen Prozesses zurück. Sehen Sie sich dazu POSIX.1 und die getpgrp(2)-Manpage auf Ihrem POSIX-System an, um mehr Informationen über Prozessgruppen zu erhalten.

posix_setsid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Macht den aktuellen Prozess zum Prozessgruppen-Führer

int **posix_setsid** (void)

Diese Funktion macht den aktuellen Prozess zum Prozessgruppen-Führer. Sehen Sie sich dazu POSIX.1 und die setsid(2)-Manpage auf Ihrem POSIX-System an, um mehr Informationen über Prozessgruppen und Job-Kontrolle zu erhalten.

Gibt die Session-ID zurück.

posix_setpgid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Setzt die Prozessgruppenkennung (Process Group ID) für die Job-Kontrolle

int **posix_setpgid** (int pid, int pgid)

Diese Funktion fügt den Prozess *pid* der Prozessgruppe *pgid* hinzu. Sehen Sie sich dazu POSIX.1 und die setpgid(2)-Manpage auf Ihrem POSIX-System an, um mehr Informationen über Prozessgruppen und Job-Kontrolle zu erhalten.

Gibt bei Erfolg TRUE und andernfalls FALSE zurück.

posix_getpgid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die Prozessgruppenkennung (Process Group ID) für die Job-Kontrolle

int **posix_getpgid** (int pid)

Diese Funktion gibt die Prozessgruppenkennung des Prozzesses *pid* zurück.

Dies ist keine POSIX-Funktion, ist aber auf BSD- und System V-Systemen gängig. Falls Ihr System diese Funktion nicht auf der Systemebene unterstützt, gibt diese PHP-Funktion immer FALSE zurück.

posix_getsid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert die aktuelle Session-ID (sid) des Prozesses

int **posix_getsid** (int pid)

Diese Funktion gibt die Session-ID des Prozesses *pid* zurück. Falls *pid* 0 ist, wird die Session-ID des aktuellen Prozesses zurückgegeben.

Dies ist keine POSIX-Funktion, ist aber auf BSD- und System V-Systemen gängig. Falls Ihr System diese Funktion nicht auf der Systemebene unterstützt, gibt diese PHP-Funktion immer FALSE zurück.

posix_uname (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert Auskunft über das System

array **posix_uname** (void)

Diese Funktion gibt eine Reihe von Daten mit Informationen über das System zurück. Die Indizes dieser Daten sind

- sysname - Name des Betriebssystems (z.B. Linux)
- nodename - Name des Systems (z.B. valiant)
- release - Release des Betriebssystems (z.B. 2.2.10)
- version - Version des Betriebssystems (z.B. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - Systemarchitektur (z.B. i586)
- domainname - DNS Domainname (z.B. php.net)

domainname ist eine GNU-Erweiterung und nicht Teil von POSIX.1. Deshalb steht dieses Feld nur auf GNU-Systemen oder bei Verwendung der GNU libc zur Verfügung.

Posix erfordert, dass Sie bezüglich des Formats der Werte nichts voraussetzen dürfen, z.B. können Sie sich weder darauf verlassen, dass die Release-Nummer aus drei Ziffern besteht noch auf irgendetwas anderes, das von dieser Funktion zurückgegeben wird.

posix_times (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert Rechenzeiten

array **posix_times** (void)

Diese Funktion gibt eine Reihe von Zahlenwerten mit Informationen über die Auslastung der CPU durch den aktuellen Prozess zurück. Die Indizes dieser Zahlenwerte sind

- ticks - die Anzahl von Clock-Ticks, die seit dem Reboot verstrichen sind.
- utime - Benutzerzeit, die vom aktuellen Prozess verbraucht wurde.
- stime - Systemzeit, die vom aktuellen Prozess verbraucht wurde.

- cutime - Benutzerzeit, die vom aktuellen Prozess und dessen Kindprozessen verbraucht wurde.
- cstime - Systemzeit, die vom aktuellen Prozess und dessen Kindprozessen verbraucht wurde.

posix_ctermid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert den Pfad des maßgebenden Terminals

string **posix_ctermid** (void)

Muss noch geschrieben werden.

posix_ttyname (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ermittelt den Namen des Terminal-Devices

string **posix_ttyname** (int fd)

Muss noch geschrieben werden.

posix_isatty (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ermittelt, ob ein Datei-Deskriptor ein interaktives Terminal ist

bool **posix_isatty** (int fd)

Muss noch geschrieben werden.

posix_getcwd (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Pfadname des aktuellen Verzeichnisses

string **posix_getcwd** (void)

Sollte ASAP geschrieben werden.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Erzeugt eine "FIFO special"-Datei (named pipe)

bool **posix_mkfifo** (string Pfadname, int Modus)

Sollte ASAP geschrieben werden.

Anmerkung: Wenn Safe Mode aktiviert ist, überprüft PHP ob das Verzeichnis in dem das Skript ausgeführt werden soll, die gleiche UID hat wie das Skript selbst.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert zu einem Gruppennamen Informationen über diese Gruppe

array **posix_getgrnam** (string Gruppenname)

Muss noch geschrieben werden.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert zu einer Gruppen-ID Informationen über diese Gruppe

array **posix_getgrgid** (int gid)

Muss noch geschrieben werden.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert zu einem Benutzernamen Informationen über diese Benutzerin

array **posix_getpwnam** (string Benutzername)

Diese Funktion gibt ein assoziatives Array zurück, das Informationen über die Benutzerin enthält, auf die der in `Benutzername` übergebene alphanumerische Benutzername verweist.

Die Elemente des zurückgegebenen Array sind:

Tabelle 1. Das Benutzerinformation-Array

Element	Beschreibung
name	Das name-Element enthält den Benutzernamen. Es handelt sich dabei nicht um ihren wirklichen, kompletten Namen, sondern um einen kurzen "Bezeichner" der Benutzerin mit normalerweise weniger als 16 Zeichen. Der Benutzername sollte der selbe wie der bei Aufruf der Funktion verwendete Parameter `Benutzername` sein und ist von daher redundant.

Element	Beschreibung
passwd	Das passwd-Element enthält das verschlüsselte Passwort der Benutzerin. Oft, zum Beispiel wenn auf einem System "shadow"-Passwörter verwendet werden, wird stattdessen ein Sternchen zurückgegeben.
uid	Die Benutzer-ID in numerischem Format.
gid	Die Gruppen-ID der Benutzerin. Benutzen Sie die Funktion posix_getgrgid(), um den Gruppennamen und eine Liste der Gruppenmitglieder aufzulösen.
gecos	GECOS ist ein veralteter Begriff, der sich auf das finger-Informationsfeld auf einem Honeywell Stapelverarbeitungssystem bezieht. Das Feld gibt es aber immer noch und sein Inhalt wurde durch POSIX formalisiert. Es enthält eine durch Komma getrennte Liste, bestehend aus dem kompletten Namen der Benutzerin, der Telefonnummer des Büros, der Zimmernummer des Büros und der privaten Telefonnummer. Auf den meisten Systemen ist nur der komplette Name der Benutzerin verfügbar.
dir	Dieses Element enthält den absoluten Pfad des Homeverzeichnisses der Benutzerin.
shell	Das shell-Element enthält den absoluten Pfad zur standardmäßigen Shell der Benutzerin.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Liefert zu einer Benutzer-ID Informationen über diese Benutzerin

array **posix_getpwuid** (int uid)

Diese Funktion gibt ein assoziatives Array zurück, das Informationen über die Benutzerin enthält, auf die die in `uid` übergebene numerische Benutzer-ID verweist.

Die Elemente des zurückgegebenen Array sind:

Tabelle 1. Das Benutzerinformation-Array

Element	Beschreibung
name	Das name-Element enthält den Benutzernamen. Es handelt sich dabei nicht um ihren wirklichen, kompletten Namen, sondern um einen kurzen "Bezeichner" der Benutzerin mit normalerweise weniger als 16 Zeichen.

Element	Beschreibung
passwd	Das passwd-Element enthält das verschlüsselte Passwort der Benutzerin. Oft, zum Beispiel wenn auf einem System "shadow"-Passwörter verwendet werden, wird stattdessen ein Sternchen zurückgegeben.
uid	Die Benutzer-ID sollte die selbe wie der Parameter `uid` sein und ist von daher redundant.
gid	Die Gruppen-ID der Benutzerin. Benutzen Sie die Funktion posix_getgrgid(), um den Gruppennamen und eine Liste der Gruppenmitglieder aufzulösen.
gecos	GECOS ist ein veralteter Begriff, der sich auf das finger-Informationsfeld auf einem Honeywell Stapelverarbeitungssystem bezieht. Das Feld gibt es aber immer noch und sein Inhalt wurde durch POSIX formalisiert. Es enthält eine durch Komma getrennte Liste, bestehend aus dem kompletten Namen der Benutzerin, der Telefonnummer des Büros, der Zimmernummer des Büros und der privaten Telefonnummer. Auf den meisten Systemen ist nur der komplette Name der Benutzerin verfügbar.
dir	Dieses Element enthält den absoluten Pfad des Homeverzeichnisses der Benutzerin.
shell	Das shell-Element enthält den absoluten Pfad zur standardmäßigen Shell der Benutzerin.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Liefert Informationen über Systemressourcen-Limits

array **posix_getrlimit** (void)

Sollte ASAP geschrieben werden.

LXXVIII. PostgreSQL Funktionen

Postgres, ursprünglich entwickelt im UC Berkeley Computer Science Department, hat Pionierarbeit bei objektrelationalen Datenbankkonzepten geleistet, die jetzt Einzug in kommerzielle Datenbanken halten. Postgres bietet SQL92/SQL3 Sprachunterstützung, Transaktionen und erweiterbare Typenklassen. PostgreSQL ist eine Public Domain und Open Source Weiterentwicklung des ursprünglichen Berkeley-Codes.

PostgreSQL ist kostenlos. Die aktuelle Version ist erhältlich bei www.PostgreSQL.org (http://www.postgresql.org/).

Seit Version 6.3 (03/02/1998) benutzt PostgreSQL Unix Domain Sockets, siehe folgende Tabelle. Der Socket ist in `/tmp/.s.PGSQL.5432` zu finden. Der Schalter -i, der dem **postmaster** mitgegeben werden kann, weist diesen an, sowohl über TCP/IP als auch über UNIX Domain Sockets eine Verbindung bereitzustellen.

Tabelle 1. Postmaster und PHP

Postmaster	PHP	Status
postmaster &	pg_connect("dbname=MyDbName");	OK
postmaster -i &	pg_connect("dbname=MyDbName");	OK
postmaster &	pg_connect("host=localhost dbname=MyDbName");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at ’localhost’ on port ’5432’? in /path/to/file.php3 on line 20.
postmaster -i &	pg_connect("host=localhost dbname=MyDbName");	OK

Eine Verbindung läßt sich auch mit folgendem Befehl herstellen: **$conn = pg_Connect("host=myHost port=myPort tty=myTTY options=myOptions user=myUser password=myPassword dbname=myDB");**

Die bisherige Syntax: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** ist veraltet und sollte nicht weiter verwendet werden.

Um die Large Object-Schnittstelle zu benutzen, ist es nötig, diese in einem Transaktionsblock einzuschließen. Ein Transaktionsblock beginnt mit einem **begin** und endet, wenn die Transaktion gültig war, mit **commit** oder **end**. Wenn die Transaktion fehlschlägt, sollte sie mit **abort** oder **rollback** geschlossen werden.

Beispiel 1. Grosse Objekte benutzen

```
<?php
    $database = pg_Connect ("", "", "", "", "jacarta");
    pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
```

```
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
    pg_exec ($database, "commit")
?>
```

pg_Close (PHP 3, PHP 4 >= 4.0.0)

Schließt eine PostgreSQL-Verbindung

bool **pg_close** (int connection)

Liefert `FALSE` zurück, wenn die Verbindung zur Datenbank keine gültige Kennung hat, ansonsten ist der Rückgabewert `TRUE`. Schließt eine PostgreSQL Datenbank-Verbindung mit der angegebenen Verbindungskennung.

pg_cmdTuples (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl betroffener Tupel zurück

int **pg_cmdtuples** (int result_id)

pg_cmdTuples() gibt die Anzahl der Tupel (instances) einer INSERT, UPDATE oder DELETE Abfrage zurück. Die Funktion liefert 0 zurück, wenn die Abfrage keinen Tupel liefert.

Beispiel 1. pg_cmdtuples

```
<?php
$result = pg_exec($conn, "INSERT INTO verlag VALUES ('Autor')");
$cmdtuples = pg_cmdtuples($result);
echo $cmdtuples . " <- cmdtuples affected.";
?>
```

pg_Connect (PHP 3, PHP 4 >= 4.0.0)

Öffnet eine Verbindung

int **pg_connect** (string conn_string)

Öffnet eine Verbindung zu einer PostgreSQL-Datenbank und liefert bei Erfolg die Verbindungskennung zurück - oder `FALSE`, falls keine Verbindung hergestellt werden konnte. Alle Verbindungsdaten sind in dem String *`conn_string`* zusammengefasst.

Beispiel 1. Gebrauch von pg_connect()

```
<?php
$dbconn = pg_Connect ("dbname=mary");
//connect to a database named "mary"
$dbconn2 = pg_Connect ("host=localhost port=5432 dbname=mary");
//connect to a database named "mary" on "localhost" at port "5432"
$dbconn3 = pg_Connect ("user=lamb passwd=baaaa dbname=mary ");
//connect to a database named "mary" with a username and password
?>
```

Die von dieser Funktion zurückgelieferte Verbindungskennung wird von anderen PostgreSQL-Funktionen benötigt. Es ist auch möglich, mehrere Verbindungen gleichzeitig zu öffnen.

Die möglichen Argumente der Verbindungsdaten sind `dbname` `port`, `host`, `tty`, `options`, `user`, und `password`.

Die alte Syntax int **pg_connect** (string host, string port, string options, string tty, string dbname) sollte nicht mehr verwendet werden.

Siehe auch **pg_pConnect()**.

pg_DBname (PHP 3, PHP 4 >= 4.0.0)

Name der Datenbank

string **pg_dbname** (int connection)

Gibt den Namen der Datenbank, deren Verbindungskennung übergeben wurde, aus - oder `FALSE`, wenn es sich um eine ungültige Kennung handelt.

pg_end_copy (PHP 4 >= 4.0.3)

Syncronisation mit dem PostgreSQL backend

bool **pg_end_copy** ([resource connection])

pg_end_copy() syncronisiert das PostgreSQL frontend mit dem backend nach einer Kopier-Operation. Die Funktion muss aufgerufen werden, damit das backend mit dem frontend synchronisiert bleibt. Liefert `TRUE` im Erfolgsfall, sonst `FALSE`.

Weiter Details und ein Beispiel finden Sie bei der Funktion pg_put_line().

pg_ErrorMessage (PHP 3, PHP 4 >= 4.0.0)

Fehlermeldung

string **pg_errormessage** (int connection)

Liefert einen String, der die Fehlermeldung enthält oder `FALSE` bei Versagen. Details über den Fehler können aber eher nicht durch pg_errormessage bezogen werden. Wenn der Fehler bei der letzten Datenbank-Aktion, für die eine gültige Kennung existierte, auftrat, liefert die Funktion einen String mit der Fehlermeldung des Backend-Servers aus.

pg_Exec (PHP 3, PHP 4 >= 4.0.0)

Führt eine Abfrage aus

int **pg_exec** (int connection, string query)

Gibt eine Ergebniskennung zurück, wenn die Abfrage ausgeführt wurde oder FALSE bei gescheiterter Abfrage oder ungültiger Verbindungskennung. (Details über den Fehler kann man bei gültiger Verbindung durch den Gebrauch von **pg_ErrorMessage()** erhalten). Diese Funktion sendet ein SQL-Statement an die durch die Verbindungskennung spezifizierte PostgreSQL Datenbank. Die Verbindung muß eine gültige, durch **pg_Connect()** zurückgegebene Kennung haben. Der Rückgabewert dieser Funktion ist eine Kennung, die dazu verwendet kann, das Abfrageergebnis mit anderen PostgreSQL-Funktionen anzusprechen.

Anmerkung: PHP/FI lieferte als Rückgabewert 1, wenn die Abfrage kein Ergebnis liefern würde (beispielsweise INSERTs oder UPDATEs), und sogar einen Wert größer 1 bei SELECTs, die nichts zurückgaben. Dies trifft für PHP nicht mehr zu.

pg_Fetch_Array (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Holt eine Datenbankreihe als Array

array **pg_fetch_array** (int result, int row [, int result_type])

Rückgabewert: Ein Array, das mit der eingelesenen Reihe identisch ist, oder FALSE, wenn keine Reihe mehr gelesen werden kann.

pg_fetch_array() ist eine erweiterte Version von pg_fetch_row(). Zusätzlich zur Speicherung der Daten in einem aufgezählten Array speichert diese Funktion die Daten in assoziativen Indexen, indem es den Feldnamen als Schlüssel verwendet.

Das dritte optionale Argument *result_type* in **pg_fetch_array()** ist eine Konstante, die folgende Werte annehmen kann: PGSQL_ASSOC, PGSQL_NUM und PGSQL_BOTH.

Anmerkung: *Result_type* wurde in PHP 4.0 hinzugefügt.

Wichtig zu bemerken ist, daß **pg_fetch_array()** nicht wesentlich langsamer als pg_fetch_row() ist, obwohl eine umfassendere Funktion zur Verfügung gestellt wird.

Für weitere Einzelheiten siehe auch pg_fetch_row()

Beispiel 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("","","","","verlag");
if (!$conn) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM autoren");
if (!$result) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
```

```
}

$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n";
$arr = pg_fetch_array ($result, 1);
echo $arr["autor"] . " <- array\n";
?>
```

pg_Fetch_Object (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Holt einen Datensatz als Objekt

object **pg_fetch_object** (int result, int row [, int result_type])

Rückgabewert: Ein Objekt, dessen Eigenschaften mit denen der Felder des erhaltenen Datensatzes übereinstimmen oder `FALSE`, wenn kein Datensatz mehr zurückgegeben wird.

pg_fetch_object() ist pg_fetch_array() ähnlich, mit einem Unterschíed: Statt eines Arrays wird ein Objekt zurückgegeben. Indirekt heißt das, dass man die Daten nur über die Feldnamen und nicht über ihre Offsets ansprechen kann (Zahlen sind nicht erlaubte Eigenschaftsnamen).

Das dritte, optionale Argument *`result_type`* in **pg_fetch_object()** ist eine Konstante und kann die folgenden Werte annehmen: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH.

Anmerkung: *`Result_type`* wurde in PHP 4.0 hinzugefügt

Von der Geschwindigkeit her ist die Funktion mit pg_fetch_array() identisch und fast so schnell wie pg_fetch_row() (Der Unterschied ist unwesentlich).

Siehe auch: pg_fetch_array() und pg_fetch_row().

Beispiel 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
    <H3>Konnte keine Datenbankverbindung zu
    <? echo $database ?> herstellen.</H3> <?
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>
```

```
<PRE><?php
$fields[] = Array ("autor", "Autor");
$fields[] = Array ("jahr",  " Jahr");
$fields[] = Array ("titel", " Titel");
$row= 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)):
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "----------\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
?>
```

pg_Fetch_Row (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Holt einen Datensatz als numerisches Array

array **pg_fetch_row** (int result, int row)

Rückgabewert: Ein Array, das mit dem gelesenen Datensatz übereinstimmt oder `FALSE`, wenn kein Datensatz mehr zurückgegeben wird.

pg_fetch_row() holt eine Datensatz aus dem duch die Kennung festgelegten Ergebnis. Der Datensatz wird als numerisches Array zurückgegeben. Jedes Ergebnisfeld wird in einem Array-Offset gespeichert, angefangen bei Offset 0.

Ein nachfolgender Aufruf von **pg_fetch_row()** wird den nächste Datensatz im Ergebnissatz ausgeben, oder `FALSE`, wenn keine auszugebender Datensatz mehr vorhanden ist.

Siehe auch: pg_fetch_array(), pg_fetch_object(), pg_result().

Beispiel 1. Postgres fetch row

```
<?php
 $conn = pg_pconnect("","","","","verlag");
if (!$conn) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM autoren");
if (!$result) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
```

```
}

$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n";
?>
```

pg_FieldIsNull (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob ein Feld `NULL` ist

int **pg_fieldisnull** (int result_id, int row, mixed field)

Prüft, ob ein Feld `NULL` ist oder nicht. Die Funktion liefert 0 zurück, wenn das Feld in der übergebenen Reihe nicht `NULL` ist und 1, wenn das Feld `NULL` ist. Das zu prüfende Feld kann als Nummer oder Feldname angegeben werden. Die Numerierung der Felder eines Datensatzes beginnt bei 0.

pg_FieldName (PHP 3, PHP 4 >= 4.0.0)

Gibt den Namen eines Feldes zurück

string **pg_fieldname** (int result_id, int field_number)

pg_FieldName() gibt den Feldnamen des Feldes, das zu der übergebenen Feldnummer gehört, aus dem durch die Kennung zu definierenden Ergebnis aus. Die Feldnummerierung beginnt bei 0.

pg_FieldNum (PHP 3, PHP 4 >= 4.0.0)

Liefert die Feldnummer eines Feldes

int **pg_fieldnum** (int result_id, string field_name)

pg_FieldNum() gibt die Nummer der Spalte aus, die zu dem übergebenen Feldnamen in dem duch die Kennung definierten Ergebnis gehört. Die Feldnumerierung beginnt bei 0. Diese Funktion gibt bei einem Fehler -1 aus.

pg_FieldPrtLen (PHP 3, PHP 4 >= 4.0.0)

Liefert die angegebene Feldlänge

int **pg_fieldprtlen** (int result_id, int row_number, string field_name)

pg_FieldPrtLen() gibt die aktuelle Länge (Anzahl der Zeichen) eines bestimmten Wertes in einem Postgres SQL-Ergebnis aus. Die Feldnumerierung beginnt bei 0. Die Funktion liefert bei einem Fehler -1 zurück.

pg_FieldSize (PHP 3, PHP 4 >= 4.0.0)

Liefert die interne Speichergröße des benannten Felds

int **pg_fieldsize** (int result_id, int field_number)

pg_FieldSize() liefert die interne Speichergröße (in Bytes) des angegebenen Feldes in dem durch die Kennung definierten Ergebnis. Die Feldnumerierung beginnt bei 0. Eine Feldgröße von -1 deutet auf ein Feld mit variabler Länge hin. Diese Funktion gibt im Fehlerfall `FALSE` zurück.

pg_FieldType (PHP 3, PHP 4 >= 4.0.0)

Liefert den Feldtyp der entsprechenden Feldnummer

string **pg_fieldtype** (int result_id, int field_number)

pg_FieldType() liefert einen String, der den Feldtyp des bezeichneten Feldes aus einem definierten Ergebnis enthält. Die Feldnumerierung beginnt bei 0.

pg_FreeResult (PHP 3, PHP 4 >= 4.0.0)

Gibt durch Ergebnisse belegten Speicher frei

int **pg_freeresult** (int result_id)

pg_FreeResult() braucht nur aufgerufen werden, wenn man befürchten muss, dass während des Skriptablaufes zu viel Speicher verbraucht wird. Der Speicher, den Ergebnisse belegen, wird automatisch nach Beendigung des Skriptes freigegeben. Wer allerdings sicher ist, die Ergebnisdaten innerhalb des Skriptes nicht mehr zu benötigen, kann **pg_FreeResult()** mit der Ergebniskennung als Argument aufrufen, und der entsprechende Speicher wird freigegeben.

pg_GetLastOid (PHP 3, PHP 4 >= 4.0.0)

Gibt die letzte Objektkennung aus

int **pg_getlastoid** (int result_id)

pg_GetLastOid() kann verwendet werden, um die einem eingefügten Tupel zugeordnete OID zu erhalten, wenn die Ergebniskennung vom letzten mit **pg_Exec()** ausgeführten Befehls benutzt wird und dieser ein SQL INSERT war. Die Funktion gibt ein positives Integer zurück, wenn eine gültige OID existiert. -1 wird zurückgegeben, wenn ein Fehler aufgetreten ist oder der letzte mit **pg_Exec()** abgesetzte Befehl kein INSERT war.

pg_Host (PHP 3, PHP 4 >= 4.0.0)

Gibt den Hostnamen zurück

string **pg_host** (int connection_id)

pg_Host() liefert den Hostnamen des Datenbankservers, der durch die übergebene Kennung verbunden ist.

pg_loclose (PHP 3, PHP 4 >= 4.0.0)

Schließt ein großes Objekt

void **pg_loclose** (int fd)

pg_loclose() schließt ein Inversion Large Object. `fd` ist der Dateibeschreiber des mit pg_loopen() geöffneten Objekts.

pg_locreate (PHP 3, PHP 4 >= 4.0.0)

Erstellt ein großes Objekt

int **pg_locreate** (int conn)

pg_locreate() erstellt ein Inversion Large Object und liefert dessen OID zurück. `conn` bezeichnet eine gültige Datenbankverbindung. Die PostgresSQL Zugriffsmöglichkeiten INV_READ, INV_WRITE und INV_ARCHIVE werden nicht unterstützt, das Objekt wird stets mit Lese- und Schreibzugriff erstellt. INV_ARCHIVE wird seit Version 6.3 und höher von PostgresSQL selbst nicht mehr unterstützt.

pg_loexport (PHP 4)

export a large object to file

bool **pg_loexport** (int oid, int file [, int connection_id])

The *oid* argument specifies the object id of the large object to export and the *filename* argument specifies the pathname of the file. Returns FALSE if an error occurred, TRUE otherwise. Remember that handling large objects in PostgreSQL must happen inside a transaction.

pg_loimport (PHP 4)

import a large object from file

int **pg_loimport** (int file [, int connection_id])

The *filename* argument specifies the pathname of the file to be imported as a large object. Returns FALSE if an error occurred, object id of the just created large object otherwise. Remember that handling large objects in PostgreSQL must happen inside a transaction.

pg_loopen (PHP 3, PHP 4 >= 4.0.0)

Öffnet ein grosses Objekt

int **pg_loopen** (int conn, int objoid, string mode)

pg_loopen() öffnet ein Inversion Large Object und liefert einen Dateibeschreiber des großen Objektes. Dieser beinhaltet Informationen über die Verbindung. Der Dateibeschreiber des großen Objektes sollte daher geschlossen werden, bevor die Verbindung beendet wird. *objoid* definiert die gültige OID eines grossen Objektes und *mode* kann "r", "w" oder "rw" sein.

pg_loread (PHP 3, PHP 4 >= 4.0.0)

Liest ein grosses Objekt

string **pg_loread** (int fd, int len)

pg_loread() liest höchstens *len* Bytes eines grossen Objektes und gibt dies als String zurück. *fd* legt einen gültigen Dateibeschreiber fest und *len* definiert die maximal erlaubte Grösse des Segmentes des großen Objektes.

pg_loreadall (PHP 3, PHP 4 >= 4.0.0)

Liest ein grosses Objekt vollständig

void **pg_loreadall** (int fd)

pg_loreadall() liest ein grosses Objekt und reicht es sofort nach dem Senden der entsprechenden Header an den Browser weiter. Wird hauptsächlich für Binärdaten wie Bilder oder Sound genutzt.

pg_lounlink (PHP 3, PHP 4 >= 4.0.0)

Ein grosses Objekt löschen

void **pg_lounlink** (int conn, int lobjid)

pg_lounlink() löscht das durch den *lobjid* Dateibeschreiber festgelegte grosse Objekt.

pg_lowrite (PHP 3, PHP 4 >= 4.0.0)

Schreibt in ein grosses Objekt

int **pg_lowrite** (int fd, string buf)

pg_lowrite() schreibt aus einer Variablen höchstens *buf* in ein grosses Objekt und liefert die Anzahl der tasächlich geschriebenen Bytes zurück, oder FALSE, wenn ein Fehler auftritt. *fd* ist der Dateibeschreiber des grossen Objektes aus pg_loopen().

pg_NumFields (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl der Felder aus

int **pg_numfields** (int result_id)

pg_NumFields() liefert die Anzahl der Felder (Spalten) eines PostgreSQL-Ergebnisses. Das Argument ist eine gültige Ergebniskennung, die von **pg_Exec()** zurückgegeben wurde. Diese Funktion gibt im Fehlerfall -1 aus.

pg_NumRows (PHP 3, PHP 4 >= 4.0.0)

Gibt die Anzahl der Zeilen aus

int **pg_numrows** (int result_id)

pg_NumRows() liefert die Anzahl der Zeilen in einem PostgreSQL-Ergebnis. Das Argument ist eine gültige Ergebniskennung, die von **pg_Exec()** zurückgegeben wurde. Diese Funktion gibt im Fehlerfall -1 aus.

pg_Options (PHP 3, PHP 4 >= 4.0.0)

Liefert Verbindungsoptionen

string **pg_options** (int connection_id)

pg_Options() liefert einen String, der die Optionen einer durch die Kennung bestimmtenn Verbindung enthält.

pg_pConnect (PHP 3, PHP 4 >= 4.0.0)

Stellt eine persistente Datenbankverbindung her

int **pg_pconnect** (string conn_string)

Öffnet eine persistente Verbindung zu einer PostgreSQL-Datenbank und liefert bei Erfolg die Verbindungskennung zurück - oder `FALSE`, falls keine Verbindung hergestellt werden konnte. Alle Verbindungsdaten sind in dem String *`conn_string`* zusammengefasst.

Die von dieser Funktion zurückgelieferte Verbindungskennung wird von anderen PostgreSQL-Funktionen benötigt. Es ist auch möglich, mehrere Verbindungen gleichzeitig zu öffnen.

Die möglichen Argumente der Verbindungsdaten sind *`dbname`* *`port`*, *`host`*, *`tty`*, *`options`*, *`user`*, und *`password`*.

Die alte Syntax int **pg_pconnect** (string host, string port, string options, string tty, string dbname) sollte nicht mehr verwendet werden.

Siehe auch pg_connect().

pg_Port (PHP 3, PHP 4 >= 4.0.0)

Gibt die Portnummer aus

int **pg_port** (int connection_id)

pg_Port() Gibt die Portnummer der durch die Kennung bezeichneten PostgresSQL-Verbindung aus.

pg_put_line (PHP 4 >= 4.0.3)

Sendet eine NULL-terminierte Zeichenkette zum PostgreSQL backend

bool **pg_put_line** ([resource connection_id, string Daten])

pg_put_line() sendet eine NULL-terminierte Zeichenkette an den PostgreSQL backend server. Nützliche ist die Funktion um bespielsweise Daten mit hoher Geschnwindigkeit in eine Datenbanktabelle einzufügen, wenn dieser Vorgang mit einer PostgreSQL copy-Operation begonnen wurde. Der abschließende NULL-character wird automatisch hinzugefügt. Liefert `TRUE` im Erfolgsfall, sonst `FALSE`.

Anmerkung: Beachten sie, dass die Anwendung explizit die beiden Zeichen "\." in der letzten Zeile senden muss, um damit dem backend das Datenende anzuzeigen.

Beispiel 1. Schnelles Einfügen von Daten in eine Tabelle

```
<?php
    $conn = pg_pconnect ("dbname=foo");
    pg_exec($conn, "create table bar (a int4, b char(16), d float8)");
```

```
    pg_exec($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

Siehe auch pg_end_copy().

pg_Result (PHP 3, PHP 4 >= 4.0.0)

Liefert Werte eines bezeichneten Ergebnisses

mixed **pg_result** (int result_id, int row_number, mixed fieldname)

pg_Result() liefert Werte aus einem durch **pg_Exec()** produzierten Ergebnis, dessen Kennung übergeben werden muß. *row_number* und *fieldname* geben an, welche Zelle der Ergebnistabelle zurückgegeben werden soll. Die Zeilennumerierung beginnt bei 0. Anstatt des Feldnamens kann auch der Feldindex als Zahl ohne Anführungszeichen angegeben werden. Der Feldindex beginnt bei 0.

PostgreSQL hat viele eingebaute Typen, und nur die grundlegenden werden hier direkt unterstützt. Alle Formen von Interger, Boolean und OID-Typen werden als Integer-Werte zurückgegeben, alle Formen von float und real als double. Alle anderen Formen, einschließlich der Arrays, werden als Strings in der durch den **psql**-Befehl bekannten PostgreSQL-Standardformatierung zurückgeliefert.

pg_set_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Set the client encoding

int **pg_set_client_encoding** ([int connection, string encoding])

The function set the client encoding and return 0 if success or -1 if error.

encoding is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

Anmerkung: This function requires PHP-4.0.2 or higher and PostgreSQL-7.0 or higher.

The function used to be called **pg_setclientencoding()**.

See also pg_client_encoding().

pg_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Get the client encoding

string **pg_client_encoding** ([int connection])

The functions returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

Anmerkung: This function requires PHP-4.0.2 or higher and PostgreSQL-7.0 or higher.

The function used to be called **pg_clientencoding()**.

See also pg_set_client_encoding().

pg_trace (PHP 4)

Trace a connection to PostgreSQL server

bool **pg_trace** (string filename [, string mode [, int connection]])

Enables tracing of the PostgreSQL frontend/backend communication to a debugging file. To fully understand the results one needs to be familiar with the internals of PostgreSQL communication protocol. For those who are not, it can still be useful for tracing errors in queries sent to the server, you could do for example 'grep '^To backend' trace.log' and see what queriew actually were sent to the PostgreSQL server.

filename and *mode* are the same as in fopen() (*mode* defaults to 'w'), *connection* specifies the connection to trace and defaults to the last one opened.

Returns TRUE if *filename* could be opened for logging, FALSE otherwise.

Siehe auch fopen() and pg_untrace().

pg_tty (PHP 3, PHP 4 >= 4.0.0)

Gibt den tty-Namen aus

string **pg_tty** (int connection_id)

pg_tty() gibt den tty-Namen aus, an den die serverseitigen Debuggermeldungen der durch die Kennung definierten Verbindung geschickt werden.

pg_untrace (PHP 4)

Stop tracing a connection to PostgreSQL server

bool **pg_untrace** ([int connection])

Stop tracing started by pg_trace(). `connection` specifies the connection that was traced and defaults to the last one opened.

Returns always `TRUE`.

Siehe auch pg_trace().

LXXIX. Process Control Functions

Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a webserver environment and unexpected results may happen if any Process Control functions are used within a webserver environment.

This documentation is intended to explain the general usage of each of the Process Control functions. For detailed information about Unix process control you are encouraged to consult your systems documentation including fork(2), waitpid(2) and signal(2) or a comprehensive reference such as Advanced Programming in the UNIX Environment by W. Richard Stevens (Addison-Wesley).

Process Control support in PHP is not enabled by default. You will need to use the --enable-pcntl configuration option when compiling PHP to enable Process Control support.

> **Anmerkung:** Currently, this module will not function on non-Unix platforms (Windows).

The following list of signals are supported by the Process Control functions. Please see your systems signal(7) man page for details of the default behavior of these signals.

Tabelle 1. Supported Signals

SIGFPE	SIGCONT	SIGKILL
SIGSTOP	SIGUSR1	SIGTSTP
SIGHUP	SIGUSR2	SIGTTIN
SIGINT	SIGSEGV	SIGTTOU
SIGQUIT	SIGPIPE	SIGURG
SIGILL	SIGALRM	SIGXCPU
SIGTRAP	SIGTERM	SIGXFSZ
SIGABRT	SIGSTKFLT	SIGVTALRM
SIGIOT	SIGCHLD	SIGPROF
SIGBUS	SIGCLD	SIGWINCH
SIGPOLL	SIGIO	SIGPWR
SIGSYS		

Process Control Example

This example forks off a daemon process with a signal handler.

Beispiel 1. Process Control Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
```

```
     die("could not fork");
} else if ($pid) {
     exit(); // we are the parent
} else {
     // we are the child
}

// detatch from the controlling terminal
if (!posix_setsid()) {
    die("could not detach from terminal");
}

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

// loop forever performing tasks
while(1) {

    // do something interesting here

}


function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         default:
             // handle all other signals
     }

}


?>
```

pcntl_fork (PHP 4 >= 4.1.0)

Forks the currently running process

int **pcntl_fork** (void)

The **pcntl_fork()** function creates a child process that differs from the parent process only in it's PID and PPID. Please see your system's fork(2) man page for specific details as to how fork works on your system.

On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and a PHP error is raised.

Beispiel 1. pcntl_fork() Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     // we are the parent
} else {
     // we are the child
}

?>
```

See also pcntl_waitpid() and pcntl_signal().

pcntl_signal (PHP 4 >= 4.1.0)

Installs a signal handler

bool **pcntl_signal** (int signo, mixed handle)

The **pcntl_signal()** function installs a new signal handler for the signal indicated by *signo*. The signal handler is set to *handler* which may be the name of a user created function, or either of the two global constants SIG_IGN or SIG_DFL.

pcntl_signal() returns TRUE on success or FALSE on failure.

Beispiel 1. pcntl_signal() Example

```
<?php

// signal handler function
function sig_handler($signo) {
```

```
    switch($signo) {
        case SIGTERM:
            // handle shutdown tasks
            exit;
            break;
        case SIGHUP:
            // handle restart tasks
            break;
        case SIGUSR1:
            print "Caught SIGUSR1...\n";
            break;
        default:
            // handle all other signals
    }

}


print "Installing signal handler...\n";

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

print "Generating signal SIGTERM to self...\n";

// send SIGUSR1 to current process id
posix_kill(posix_getpid(), SIGUSR1);

print "Done\n"

?>
```

See also pcntl_fork() and pcntl_waitpid().

pcntl_waitpid (PHP 4 >= 4.1.0)

Waits on or returns the status of a forked child

int **pcntl_waitpid** (int pid, int status, int options)

The **pcntl_waitpid()** function suspends execution of the current process until a child as specified by the *pid* argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child as requested by *pid* has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's waitpid(2) man page for specific details as to how waitpid works on your system.

pcntl_waitpid() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was used and no child was available

The value of *pid* can be one of the following:

Tabelle 1. possible values for *pid*

`< -1`	wait for any child process whose process group ID is equal to the absolute value of *pid*.
`-1`	wait for any child process; this is the same behaviour that the wait function exhibits.
`0`	wait for any child process whose process group ID is equal to that of the calling process.
`> 0`	wait for the child whose process ID is equal to the value of *pid*.

pcntl_waitpid() will store status information in the *status* parameter which can be evaluated using the following functions: pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

The value of *options* is the value of zero or more of the following two global constants `OR`'ed together:

Tabelle 2. possible values for *options*

`WNOHANG`	return immediately if no child has exited.
`WUNTRACED`	return for children which are stopped, and whose status has not been reported.

See also pcntl_fork(), pcntl_signal(), pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

pcntl_wexitstatus (PHP 4 >= 4.1.0)

Returns the return code of a terminated child

int **pcntl_wexitstatus** (int status)

Returns the return code of a terminated child. This function is only useful if pcntl_wifexited() returned `TRUE`.

The parameter *status* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifexited().

pcntl_wifexited (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a successful exit

int **pcntl_wifexited** (int status)

Returns `TRUE` if the child status code represents a successful exit.

The parameter `status` is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wexitstatus().

pcntl_wifsignaled (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a termination due to a signal

int **pcntl_wifsignaled** (int status)

Returns `TRUE` if the child process exited because of a signal which was not caught.

The parameter `status` is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_signal().

pcntl_wifstopped (PHP 4 >= 4.1.0)

Returns `TRUE` if child process is currently stopped

int **pcntl_wifstopped** (int status)

Returns `TRUE` if the child process which caused the return is currently stopped; this is only possible if the call to pcntl_waitpid() was done using the option `WUNTRACED`.

The parameter `status` is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid().

pcntl_wstopsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to stop

int **pcntl_wstopsig** (int status)

Returns the number of the signal which caused the child to stop. This function is only useful if pcntl_wifstopped() returned `TRUE`.

The parameter `status` is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifstopped().

pcntl_wtermsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to terminate

int **pcntl_wtermsig** (int status)

Returns the number of the signal that caused the child process to terminate. This function is only useful if pcntl_wifsignaled() returned `TRUE`.

The parameter `status` is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid(), pcntl_signal() and pcntl_wifsignaled().

pcntl_exec (PHP 4 CVS only)

Executes specified program in current process space

bool **pcntl_exec** (string path [, array args [, array envs]])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LXXX. Funktionen zur Programmausführung

Diese Funktionen stellen Mittel und Wege zur Verfügung, Befehle und externe Programme auf dem System selbst auszuführen und diese Ausführung sicher durchzuführen. Diese Funktionen stehen in enger Beziehung zum Backtick-Operator.

escapeshellarg (PHP 4 >= 4.0.3)

Maskiert eine Zeichenkette (String), um sie als Shell-Argument benutzen zu können

string **escapeshellarg** (string arg)

escapeshellarg() fügt einfache Anführungszeichen um eine Zeichenkette herum ein und maskiert alle existierenden einfachen Anführungszeichen innerhalb der Zeichenkette. Dieses erlaubt es Ihnen, eine Zeichenkette direkt an eine Shell-Funktion zu übergeben, so dass diese als eine einzige, sichere Anweisung ausgeführt werden kann. Die Funktion sollte verwendet werden, um Parameter, die aus Benutzereingaben resultieren, an Shell-Funktionen zu übergeben. Die Shell-Funktionen umfassen exec(), system() und den backtick operator. Beispiel:

```
system("ls ".escapeshellarg($dir));
```

Siehe auch:exec(), popen(), system() und den Backtick Operator.

escapeshellcmd (PHP 3, PHP 4 >= 4.0.0)

Maskiert Shell-Metazeichen

string **escapeshellcmd** (string befehl)

escapeshellcmd() maskiert alle möglichen Zeichen in einer Zeichenkette, die dazu benutzt werden könnten, um einen Shellbefehl zur Durchführung von willkürlichen Befehlen zu veranlassen. Diese Funktion sollte verwendet werden um sicherzustellen, dass alle Daten aus einer Benutzereingabe maskiert werden bevor dieses Daten zu einer der Funktionen exec(), system() oder dem Backtick Operator übergeben werden. Beispiel:

```
$e = escapeshellcmd($userinput);
system("echo $e"); // hier ist es egal, ob $e Leerstellen hat
$f = escapeshellcmd($filename);
system("touch \"/tmp/$f\"; ls -l \"/tmp/$f\""); // hier ist es nicht egal
```

Siehe auch: escapeshellarg(), exec(), popen(), system() und den Backtick Operator.

exec (PHP 3, PHP 4 >= 4.0.0)

Führt ein externes Programm aus

string **exec** (string befehl [, string array [, int return_var]])

exec() führt ein gegebenen `Befehl` aus, ohne eine Ausgabe zu erzeugen. Die Funktion gibt lediglich die letzte Zeile aus dem Befehlsergebnis zurück. Wenn Sie auf die direkte Rückgabe/ Ausgabe eines Befehls angewiesen sind, dann benutzen Sie stattdessen die Funktion passthru().

Ist der Parameter `array` angegeben, wird dieses mit jeder Zeile des Befehlsausgabe gefüllt. Beachten Sie, dass wenn das Array schon Elemente enthält, die Funktion **exec()** die Ausgabe an das Array anhängt. Wenn Sie dieses nicht wünschen, rufen Sie die Funktion **unset ()** für das Array auf, bevor Sie es als Parameter der Funktion **exec()** übergeben.

Ist der Parameter `return_var` angegeben, so wird der Rückgabestatus des ausgeführten Befehls in diese Variable geschrieben.

Beachten Sie, dass wenn Sie Benutzereingaben an diese Funktion übergeben, diese mit escapeshellcmd() behandelt, um sicherzustellen, dass niemand mit willkürlichen Befehlen Ihrem System Schaden zufügen kann.

Beachten Sie auch, dass wenn Sie ein Programm mit dieser Funktion starten und Sie es im Hintergrund laufen lassen möchten, Sie sicherstellen müssen, dass die Ausgabe des Programms in eine Datei oder anderenen Ausgabestream umgeleitet wird. Andernfalls wird PHP solange laufen, bis das Programm beendet ist.

Siehe auch: system(), passthru(), popen(), escapeshellcmd() und den Backtick Operator.

passthru (PHP 3, PHP 4 >= 4.0.0)

Führt ein externes Programm aus und zeigt dessen Ausgabe an

void **passthru** (string befehl [, int return_var])

Die Funktion **passthru()** ähnelt der Funktion exec(), da sie ebenfalls einen `Befehl` ausführt. Ist der Parameter `return_var` angegeben, wird der Rückgabestatus des UNIX-Befehls hier abgelegt. Sie sollten diese Funktion jedoch an Stelle von exec() bzw. system() benutzen, wenn es sich bei der Ausgabe des Unix-Befehls um binäre Daten handelt, welche direkt zum Browser zurückgeschickt werden müssen. Ein praktisches Beispiel hierfür ist die Ausführung des pbmplus Utility, welches direkt einen Bildstream ausgibt. Wenn Sie den Content-Type auf *image/gif* setzen, und dann das pbmplus Program aufrufen um ein GIF zu erzeugen, können Sie auf diesem Weg mittels PHP direkt ein Bild erzeugen und es ausgeben.

Beachten Sie, dass wenn Sie ein Programm mit dieser Funktion starten und Sie es im Hintergrund laufen lassen möchten, Sie sicherstellen müssen, dass die Ausgabe des Programms in eine Datei oder anderenen Ausgabestream umgeleitet wird. Andernfalls wird PHP solange laufen, bis das Programm beendet ist.

Siehe auch: exec(), system(), popen(), escapeshellcmd() und den Backtick Operator.

system (PHP 3, PHP 4 >= 4.0.0)

Führt ein externes Programm aus und zeigt dessen Ausgabe an.

string **system** (string befehl [, int return_var])

Die Funktion **system()** ähnelt der C Version der Funktion sehr, indem es einen übergebenen `Befehl` ausführt und dessen Ausgabe anzeigt. Wir als zweiter Parameter der Funktion eine Variable übergeben, so wird der Rückgabestatus des Befehls in diese geschrieben.

Beachten Sie, dass wenn Sie Benutzereingaben an diese Funktion übergeben, diese mit escapeshellcmd() behandelt, um sicherzustellen, dass niemand mit willkürlichen Befehlen Ihrem System Schaden zufügen kann.

Beachten Sie auch, dass wenn Sie ein Programm mit dieser Funktion starten und Sie es im Hintergrund laufen lassen möchten, Sie sicherstellen müssen, dass die Ausgabe des Programms in eine Datei oder anderenen Ausgabestream umgeleitet wird. Andernfalls wird PHP solange laufen, bis das Programm beendet ist.

Die **system()** Funktionsaufrufe versuchen übrigens automatisch nach jeder Ausgabezeile den Ausgabebuffer des Webservers zu flushen, wenn PHP als Servermodul läuft.

Gibt die letzte Zeile der Befehlsausgabe bei Erfolg, andernfalls `FALSE` zurück.

Wenn Sie auf die direkte Rückgabe/ Ausgabe eines Befehls angewiesen sind, dann benutzen Sie stattdessen die Funktion passthru().

Siehe auch: exec(), passthru(), popen(), escapeshellcmd() und den Backtick Operator.

LXXXI. Printer functions

These functions are only available under Windows 9.x, ME, NT4 and 2000. They have been added in PHP 4 (4.0.4).

printer_open (unknown)

Open connection to a printer

mixed **printer_open** ([string devicename])

This function tries to open a connection to the printer *`devicename`*, and returns a handle on success or `FALSE` on failure.

If no parameter was given it tries to open a connection to the default printer (if not specified in `php.ini` as `printer.default_printer`, php tries to detect it).

printer_open() also starts a device context.

Beispiel 1. printer_open() example

```
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
```

printer_abort (unknown)

Deletes the printer's spool file

void **printer_abort** (resource handle)

This function deletes the printers spool file.

`handle` must be a valid handle to a printer.

Beispiel 1. printer_abort() example

```
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
```

printer_close (unknown)

Close an open printer connection

void **printer_close** (resource handle)

This function closes the printer connection. **printer_close()** also closes the active device context.

`handle` must be a valid handle to a printer.

Beispiel 1. printer_close() example

```
$handle = printer_open();
printer_close($handle);
```

printer_write (unknown)

Write data to the printer

bool **printer_write** (resource handle, string content)

Writes *content* directly to the printer, and returns TRUE on success or FALSE if it failed.

handle must be a valid handle to a printer.

Beispiel 1. printer_write() example

```
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
```

printer_list (unknown)

Return an array of printers attached to the server

array **printer_list** (int enumtype [, string name [, int level]])

The function enumerates available printers and their capabilities. *level* sets the level of information request. Can be 1,2,4 or 5. *enumtype* must be one of the following predefined constants:

- *PRINTER_ENUM_LOCAL*: enumerates the locally installed printers.
- *PRINTER_ENUM_NAME*: enumerates the printer of *name*, can be a server, domain or print provider.
- *PRINTER_ENUM_SHARED*: this parameter can't be used alone, it has to be OR'ed with other parameters, i.e. PRINTER_ENUM_LOCAL to detect the locally shared printers.
- *PRINTER_ENUM_DEFAULT*: (Win9.x only) enumerates the default printer.
- *PRINTER_ENUM_CONNECTIONS*: (WinNT/2000 only) enumerates the printers to which the user has made connections.
- *PRINTER_ENUM_NETWORK*: (WinNT/2000 only) enumerates network printers in the computer's domain. Only valid if *level* is 1.
- *PRINTER_ENUM_REMOTE*: (WinNT/2000 only) enumerates network printers and print servers in the computer's domain. Only valid if *level* is 1.

Beispiel 1. printer_list() example

```
/* detect locally shared printer */
var_dump( printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED) );
```

printer_set_option (unknown)

Configure the printer connection

bool **printer_set_option** (resource handle, int option, mixed value)

The function sets the following options for the current connection: *handle* must be a valid handle to a printer. For *option* can be one of the following constants:

- *PRINTER_COPIES*: sets how many copies should be printed, *value* must be an integer.
- *PRINTER_MODE*: specifies the type of data (text, raw or emf), *value* must be a string.
- *PRINTER_TITLE*: specifies the name of the document, *value* must be a string.
- *PRINTER_ORIENTATION*: specifies the orientation of the paper, *value* can be either PRINTER_ORIENTATION_PORTRAIT or PRINTER_ORIENTATION_LANDSCAPE
- *PRINTER_RESOLUTION_Y*: specifies the y-resolution in DPI, *value* must be an integer.
- *PRINTER_RESOLUTION_X*: specifies the x-resolution in DPI, *value* must be an integer.
- *PRINTER_PAPER_FORMAT*: specifies the a predefined paper format, set *value* to PRINTER_FORMAT_CUSTOM if you want to specify a custom format with PRINTER_PAPER_WIDTH and PRINTER_PAPER_LENGTH. *value* can be one of the following constants.
  - *PRINTER_FORMAT_CUSTOM*: let's you specify a custom paper format.
  - *PRINTER_FORMAT_LETTER*: specifies standard letter format (8 1/2- by 11-inches).
  - *PRINTER_FORMAT_LETTER*: specifies standard legal format (8 1/2- by 14-inches).
  - *PRINTER_FORMAT_A3*: specifies standard A3 format (297- by 420-millimeters).
  - *PRINTER_FORMAT_A4*: specifies standard A4 format (210- by 297-millimeters).
  - *PRINTER_FORMAT_A5*: specifies standard A5 format (148- by 210-millimeters).
  - *PRINTER_FORMAT_B4*: specifies standard B4 format (250- by 354-millimeters).
  - *PRINTER_FORMAT_B5*: specifies standard B5 format (182- by 257-millimeter).
  - *PRINTER_FORMAT_FOLIO*: specifies standard FOLIO format (8 1/2- by 13-inch).
- *PRINTER_PAPER_LENGTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifies a custom paper length in mm, *value* must be an integer.

- *PRINTER_PAPER_WIDTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifies a custom paper width in mm, *value* must be an integer.
- *PRINTER_SCALE*: specifies the factor by which the printed output is to be scaled. the page size is scaled from the physical page size by a factor of scale/100. for example if you set the scale to 50, the output would be half of it's original size. *value* must be an integer.
- *PRINTER_BACKGROUND_COLOR*: specifies the background color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_COLOR*: specifies the text color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_ALIGN*: specifies the text alignment for the actual device context, *value* can be combined through OR'ing the following constants:
  - *PRINTER_TA_BASELINE*: text will be aligned at the base line.
  - *PRINTER_TA_BOTTOM*: text will be aligned at the bottom.
  - *PRINTER_TA_TOP*: text will be aligned at the top.
  - *PRINTER_TA_CENTER*: text will be aligned at the center.
  - *PRINTER_TA_LEFT*: text will be aligned at the left.
  - *PRINTER_TA_RIGHT*: text will be aligned at the right.

Beispiel 1. printer_set_option() example

```
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
```

printer_get_option (unknown)

Retrieve printer configuration data

mixed **printer_get_option** (resource handle, string option)

The function retrieves the configuration setting of *option*. *handle* must be a valid handle to a printer. Take a look at printer_set_option() for the settings that can be retrieved, additionally the following settings can be retrieved:

- *PRINTER_DEVICENAME* returns the devicename of the printer.
- *PRINTER_DRIVERVERSION* returns the printer driver version.

Beispiel 1. printer_get_option() example

```
$handle = printer_open();
print printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
```

printer_create_dc (unknown)

Create a new device context

void **printer_create_dc** (resource handle)

The function creates a new device context. A device context is used to customize the graphic objects of the document. *handle* must be a valid handle to a printer.

Beispiel 1. printer_create_dc() example

```
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* do some stuff with the dc */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* create another dc */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* do some stuff with the dc */

printer_delete_dc($handle);

printer_endpage($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_delete_dc (unknown)

Delete a device context

bool **printer_delete_dc** (resource handle)

The function deletes the device context and returns `TRUE` on success, or `FALSE` if an error occurred. For an example see printer_create_dc(). *`handle`* must be a valid handle to a printer.

printer_start_doc (unknown)

Start a new document

bool **printer_start_doc** (resource handle [, string document])

The function creates a new document in the printer spooler. A document can contain multiple pages, it's used to schedule the print job in the spooler. *`handle`* must be a valid handle to a printer. The optional parameter *`document`* can be used to set an alternative document name.

Beispiel 1. printer_start_doc() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_end_doc (unknown)

Close document

bool **printer_end_doc** (resource handle)

Closes a new document in the printer spooler. The document is now ready for printing. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_start_page (unknown)

Start a new page

bool **printer_start_page** (resource handle)

The function creates a new page in the active document. For an example see printer_start_doc(). *`handle`* must be a valid handle to a printer.

printer_end_page (unknown)

Close active page

bool **printer_end_page** (resource handle)

The function closes the active page in the active document. For an example see printer_start_doc(). *handle* must be a valid handle to a printer.

printer_create_pen (unknown)

Create a new pen

mixed **printer_create_pen** (int style, int width, string color)

The function creates a new pen and returns a handle to it. A pen is used to draw lines and curves. For an example see printer_select_pen(). *color* must be a color in RGB hex format, i.e. "000000" for black, *width* specifies the width of the pen whereas *style* must be one of the following constants:

- *PRINTER_PEN_SOLID*: creates a solid pen.
- *PRINTER_PEN_DASH*: creates a dashed pen.
- *PRINTER_PEN_DOT*: creates a dotted pen.
- *PRINTER_PEN_DASHDOT*: creates a pen with dashes and dots.
- *PRINTER_PEN_DASHDOTDOT*: creates a pen with dashes and double dots.
- *PRINTER_PEN_INVISIBLE*: creates an invisible pen.

printer_delete_pen (unknown)

Delete a pen

bool **printer_delete_pen** (resource handle)

The function deletes the selected pen. For an example see printer_select_pen(). It returns `TRUE` on success, or `FALSE` otherwise. *handle* must be a valid handle to a pen.

printer_select_pen (unknown)

Select a pen

void **printer_select_pen** (resource printer_handle, resource pen_handle)

The function selects a pen as the active drawing object of the actual device context. A pen is used to draw lines and curves. I.e. if you draw a single line the pen is used. If you draw an rectangle the pen is used to draw the borders, while the brush is used to fill the shape. If you haven't selected a pen

before drawing shapes, the shape won't be outlined. *printer_handle* must be a valid handle to a printer. *pen_handle* must be a valid handle to a pen.

Beispiel 1. printer_select_pen() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_brush (unknown)

Create a new brush

mixed **printer_create_brush** (int style, string color)

The function creates a new brush and returns a handle to it. A brush is used to fill shapes. For an example see printer_select_brush(). *color* must be a color in RGB hex format, i.e. "000000" for black, *style* must be one of the following constants:

- *PRINTER_BRUSH_SOLID*: creates a brush with a solid color.
- *PRINTER_BRUSH_DIAGONAL*: creates a brush with a 45-degree upward left-to-right hatch (/).
- *PRINTER_BRUSH_CROSS*: creates a brush with a cross hatch (+).
- *PRINTER_BRUSH_DIAGCROSS*: creates a brush with a 45 cross hatch (x).
- *PRINTER_BRUSH_FDIAGONAL*: creates a brush with a 45-degree downward left-to-right hatch (\ ).
- *PRINTER_BRUSH_HORIZONTAL*: creates a brush with a horizontal hatch (-).
- *PRINTER_BRUSH_VERTICAL*: creates a brush with a vertical hatch (|).
- *PRINTER_BRUSH_CUSTOM*: creates a custom brush from an BMP file. The second parameter is used to specify the BMP instead of the RGB color code.

printer_delete_brush (unknown)

Delete a brush

bool **printer_delete_brush** (resource handle)

The function deletes the selected brush. For an example see printer_select_brush(). It returns `TRUE` on success, or `FALSE` otherwise. *`handle`* must be a valid handle to a brush.

printer_select_brush (unknown)

Select a brush

void **printer_select_brush** (resource printer_handle, resource brush_handle)

The function selects a brush as the active drawing object of the actual device context. A brush is used to fill shapes. If you draw an rectangle the brush is used to draw the shapes, while the pen is used to draw the border. If you haven't selected a brush before drawing shapes, the shape won't be filled. *`printer_handle`* must be a valid handle to a printer. *`brush_handle`* must be a valid handle to a brush.

Beispiel 1. printer_select_brush() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1,1,500,500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1,501,500,1001);
printer_delete_brush($brush);

printer_delete_pen($pen);


printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_font (unknown)

Create a new font

mixed **printer_create_font** (string face, int height, int width, int font_weight, bool italic, bool underline, bool strikeout, int orientaton)

The function creates a new font and returns a handle to it. A font is used to draw text. For an example see printer_select_font(). *face* must be a string specifying the font face. *height* specifies the font height, and *width* the font width. The *font_weight* specifies the font weight (400 is normal), and can be one of the following predefined constants.

- *PRINTER_FW_THIN*: sets the font weight to thin (100).
- *PRINTER_FW_ULTRALIGHT*: sets the font weight to ultra light (200).
- *PRINTER_FW_LIGHT*: sets the font weight to light (300).
- *PRINTER_FW_NORMAL*: sets the font weight to normal (400).
- *PRINTER_FW_MEDIUM*: sets the font weight to medium (500).
- *PRINTER_FW_BOLD*: sets the font weight to bold (700).
- *PRINTER_FW_ULTRABOLD*: sets the font weight to ultra bold (800).
- *PRINTER_FW_HEAVY*: sets the font weight to heavy (900).

italic can be TRUE or FALSE, and sets whether the font should be italic.

underline can be TRUE or FALSE, and sets whether the font should be underlined.

strikeout can be TRUE or FALSE, and sets whether the font should be striked out.

orientation specifies a rotation. For an example see printer_select_font().

printer_delete_font (unknown)

Delete a font

bool **printer_delete_font** (resource handle)

The function deletes the selected font. For an example see printer_select_font(). It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a font.

printer_select_font (unknown)

Select a font

void **printer_select_font** (resource printer_handle, resource font_handle)

The function selects a font to draw text. *printer_handle* must be a valid handle to a printer. *font_handle* must be a valid handle to a font.

Beispiel 1. printer_select_font() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -
50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_logical_fontheight (unknown)

Get logical font height

int **printer_logical_fontheight** (resource handle, int height)

The function calculates the logical font height of *height*. *handle* must be a valid handle to a printer.

Beispiel 1. printer_logical_fontheight() example

```
$handle = printer_open();
print printer_logical_fontheight($handle, 72);
printer_close($handle);
```

printer_draw_roundrect (unknown)

Draw a rectangle with rounded corners

void **printer_draw_roundrect** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int width, int height)

The function simply draws a rectangle with rounded corners.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

width is the width of the ellipse.

height is the height of the ellipse.

Beispiel 1. printer_draw_roundrect() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_rectangle (unknown)

Draw a rectangle

void **printer_draw_rectangle** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)

The function simply draws a rectangle.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

Beispiel 1. printer_draw_rectangle() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
```

```
$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_elipse (unknown)

Draw an ellipse

void **printer_draw_elipse** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)

The function simply draws an ellipse. *handle* must be a valid handle to a printer.

ul_x is the upper left x coordinate of the ellipse.

ul_y is the upper left y coordinate of the ellipse.

lr_x is the lower right x coordinate of the ellipse.

lr_y is the lower right y coordinate of the ellipse.

Beispiel 1. printer_draw_elipse() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_text (unknown)

Draw text

void **printer_draw_text** (resource printer_handle, string text, int x, int y)

The function simply draws *text* at position *x*, *y* using the selected font. *printer_handle* must be a valid handle to a printer.

Beispiel 1. printer_draw_text() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial",72,48,400,false,false,false,0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_line (unknown)

Draw a line

void **printer_draw_line** (resource printer_handle, int from_x, int from_y, int to_x, int to_y)

The function simply draws a line from position *from_x*, *from_y* to position *to_x*, *to_y* using the selected pen. *printer_handle* must be a valid handle to a printer.

Beispiel 1. printer_draw_line() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
```

```
printer_close($handle);
```

printer_draw_chord (unknown)

Draw a chord

void **printer_draw_chord** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad_x, int rad_y, int rad_x1, int rad_y1)

The function simply draws an chord. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad_x is x coordinate of the radial defining the beginning of the chord.

rad_y is y coordinate of the radial defining the beginning of the chord.

rad_x1 is x coordinate of the radial defining the end of the chord.

rad_y1 is y coordinate of the radial defining the end of the chord.

Beispiel 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_pie (unknown)

Draw a pie

void **printer_draw_pie** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad1_x, int rad1_y, int rad2_x, int rad2_y)

The function simply draws an pie. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad1_x is x coordinate of the first radial's ending.

rad1_y is y coordinate of the first radial's ending.

rad2_x is x coordinate of the second radial's ending.

rad2_y is y coordinate of the second radial's ending.

Beispiel 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_bmp (unknown)

Draw a bmp

void **printer_draw_bmp** (resource handle, string filename, int x, int y)

The function simply draws an bmp the bitmap *filename* at position *x*, *y*. *handle* must be a valid handle to a printer.

The function returns `TRUE` on success, or otherwise `FALSE`.

Beispiel 1. printer_draw_bmp() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

LXXXII. Pspell Functions

These functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://pspell.sourceforge.net/ respectively, and add the `--with-pspell[=dir]` option when compiling php.

pspell_add_to_personal (PHP 4 >= 4.0.2)

Add the word to a personal wordlist

int **pspell_add_to_personal** (int dictionary_link, string word)

pspell_add_to_personal() adds a word to the personal wordlist. If you used pspell_new_config() with pspell_config_personal() to open the dictionary, you can save the wordlist later with pspell_save_wordlist(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Beispiel 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session (PHP 4 >= 4.0.2)

Add the word to the wordlist in the current session

int **pspell_add_to_session** (int dictionary_link, string word)

pspell_add_to_session() adds a word to the wordlist associated with the current session. It is very similar to pspell_add_to_personal()

pspell_check (PHP 4 >= 4.0.2)

Check a word

bool **pspell_check** (int dictionary_link, string word)

pspell_check() checks the spelling of a word and returns `TRUE` if the spelling is correct, `FALSE` if not.

Beispiel 1. pspell_check()

```
$pspell_link = pspell_new ("en");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
```

```
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_clear_session (PHP 4 >= 4.0.2)

Clear the current session

int **pspell_clear_session** (int dictionary_link)

pspell_clear_session() clears the current session. The current wordlist becomes blank, and, for example, if you try to save it with pspell_save_wordlist(), nothing happens.

Beispiel 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link);    //"Vlad" will not be saved
```

pspell_config_create (PHP 4 >= 4.0.2)

Create a config used to open a dictionary

int **pspell_config_create** (string language [, string spelling [, string jargon [, string encoding]]])

pspell_config_create() has a very similar syntax to pspell_new(). In fact, using **pspell_config_create()** immediatelly followed by pspell_new_config() will produce the exact same result. However, after creating a new config, you can also use **pspell_config_*()** functions before calling pspell_new_config() to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are ’american’, ’british’, and ’canadian’.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Beispiel 1. pspell_config_create()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_config_ignore (PHP 4 >= 4.0.2)

Ignore words less than N characters long

int **pspell_config_ignore** (int dictionary_link, int n)

pspell_config_ignore() should be used on a config before calling pspell_new_config(). This function allows short words to be skipped by the spellchecker. Words less then n characters will be skipped.

Beispiel 1. pspell_config_ignore()

```
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    //will not result in an error
```

pspell_config_mode (PHP 4 >= 4.0.2)

Change the mode number of suggestions returned

int **pspell_config_mode** (int dictionary_link, int mode)

pspell_config_mode() should be used on a config before calling pspell_new_config(). This function determines how many suggestions will be returned by pspell_suggest().

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

Beispiel 1. pspell_config_mode()

```
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
```

pspell_config_personal (PHP 4 >= 4.0.2)

Set a file that contains personal wordlist

int **pspell_config_personal** (int dictionary_link, string file)

pspell_config_personal() should be used on a config before calling pspell_new_config(). The personal wordlist will be loaded and used in addition to the standard one after you call pspell_new_config(). If the file does not exist, it will be created. The file is also the file where pspell_save_wordlist() will save personal wordlist to. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Beispiel 1. pspell_config_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_repl (PHP 4 >= 4.0.2)

Set a file that contains replacement pairs

int **pspell_config_repl** (int dictionary_link, string file)

pspell_config_repl() should be used on a config before calling pspell_new_config(). The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, pspell_store_replacement() can be used to store a replacement pair and then pspell_save_wordlist() to save the wordlist along with the replacement pairs. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Beispiel 1. pspell_config_repl()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_runtogether (PHP 4 >= 4.0.2)

Consider run-together words as valid compounds

int **pspell_config_runtogether** (int dictionary_link, bool flag)

pspell_config_runtogether() should be used on a config before calling pspell_new_config(). This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Beispiel 1. pspell_config_runtogether()

```
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, true);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_save_repl (PHP 4 >= 4.0.2)

Determine whether to save a replacement pairs list along with the wordlist

int **pspell_config_save_repl** (int dictionary_link, bool flag)

pspell_config_save_repl() should be used on a config before calling pspell_new_config(). It determines whether pspell_save_wordlist() will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if pspell_config_repl() is used, the replacement pairs will be saved by pspell_save_wordlist() anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new (PHP 4 >= 4.0.2)

Load a new dictionary

int **pspell_new** (string language [, string spelling [, string jargon [, string encoding [, int mode]]]])

pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are ’american’, ’british’, and ’canadian’.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are ’utf-8’, ’iso8859-*’, ’koi8-r’, ’viscii’, ’cp1252’, ’machine unsigned 16’, ’machine unsigned 32’. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Beispiel 1. pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_new_config (PHP 4 >= 4.0.2)

Load a new dictionary with settings based on a given config

int **pspell_new_config** (int config)

pspell_new_config() opens up a new dictionary with settings specified in a config, created with with pspell_config_create() and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by pspell_new() and pspell_new_personal().

The config parameter is the one returned by pspell_config_create() when the config was created.

Beispiel 1. pspell_new_config()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
```

pspell_new_personal (PHP 4 >= 4.0.2)

Load a new dictionary with personal wordlist

int **pspell_new_personal** (string personal, string language [, string spelling [, string jargon [, string encoding [, int mode]]]])

pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with pspell_save_wordlist(), if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using pspell_config_create(), set the personal

wordlist file with pspell_config_personal(), set the file for replacement pairs with pspell_config_repl(), and open a new dictionary with pspell_new_config().

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with '/' because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Beispiel 1. pspell_new_personal()

```
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_save_wordlist (PHP 4 >= 4.0.2)

Save the personal wordlist to a file

int **pspell_save_wordlist** (int dictionary_link)

pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with pspell_new_personal(), and the location of files to be saved specified with pspell_config_personal() and (optionally) pspell_config_repl(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Beispiel 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_store_replacement (PHP 4 >= 4.0.2)

Store a replacement pair for a word

int **pspell_store_replacement** (int dictionary_link, string misspelled, string correct)

pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by pspell_suggest() later. In order to be able to take advantage of this function, you have to use pspell_new_personal() to open the dictionary. In order to permanently save the replacement pair, you have to use pspell_config_personal() and pspell_config_repl() to set the path where to save your custom wordlists, and then use pspell_save_wordlist() for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Beispiel 1. pspell_store_replacement()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggest spellings of a word

array **pspell_suggest** (int dictionary_link, string word)

pspell_suggest() returns an array of possible spellings for the given word.

Beispiel 1. pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>";
    }
}
```

LXXXIII. GNU Readline

Die readline() Funktionen realisieren eine Schnittstelle zur GNU Readline Bibliothek. Diese Funktionen bieten die Möglichkeit Kommandozeilen zu editieren. Ein Beispiel für die Anwendung ist die Art und Weise, wie die Bash es erlaubt, die Pfeiltasten zu benutzen um Schriftzeichen einzufügen oder durch die Kommando-History zu scrollen. Auf Grund des interaktiven Charakters dieser Bibliothek ist diese von geringem Nutzen beim Programmieren von Webanwendungen, kann aber nützlich sein, wenn Sie Skripte schreiben, die dafür gedacht sind in einer Shell abzulaufen.

Die Homepage des GNU Readline Projekts finden Sie unter http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. Das Projekt wird von Chet Ramey betreut, der außerdem der Autor der Bash ist.

readline (PHP 4 >= 4.0.0)

Liest eine Zeile

string **readline** ([string prompt])

Diese Funktion gibt eine einzelne Zeichenkette vom Benutzer zurück. Sie können eine Zeichenkette angeben, mit dem Sie den Benutzer zur Eingabe auffordern. Der Zeile die Sie erhalten fehlt das abschließende Zeichen für einen Zeilenvorschub. Die Zeile müssen Sie selbst mit Hilfe der Funktion readline_add_history() der History hinzufügen.

Beispiel 1. readline()

```
//abfragen von 3 Kommandos vom Benutzer
for ($i=0; $i <; 3; $i++) {
        $zeile = readline ("Kommando: ");
        readline_add_history ($line);
}

//anzeigen der History
print_r (readline_list_history());

//anzeigen der Variablen
print_r (readline_info());
```

readline_add_history (PHP 4 >= 4.0.0)

Fügt eine Zeile zur History hinzu

void **readline_add_history** (string zeile)

Diese Funktion fügt eine Zeile zur Kommandozeilen-History hinzu.

readline_clear_history (PHP 4 >= 4.0.0)

Löscht die History

bool **readline_clear_history** (void)

Diese Funktion löscht die gesamte Kommandozeilen-History.

readline_completion_function (PHP 4 >= 4.0.0)

Registriert eine Vervollständigungsfunktion

bool **readline_completion_function** (string zeile)

Diese Funktion registriert eine Vervollständigungsfunktion. Sie müssen den Namen einer existierenden Funktion angeben, die einen Teil einer Kommandozeile akzeptiert und ein Array möglicher Übereinstimmungen zurückgibt. Das ist die gleiche Art von Funktionalität die Sie haben, wenn Sie die Tab-Taste benutzen, während Sie mit der Bash arbeiten.

readline_info (PHP 4 >= 4.0.0)

Liest/Setzt verschiedene interne readline Variablen

mixed **readline_info** ([string varname [, string neuerwert]])

Wenn diese Funktion ohne Parameter aufgerufen wird, gibt sie einen Array von Werten für alle Einstellungen zurück, die readline benutzt. Die Elemente werden durch die folgenden Werte indiziert: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name und terminal_name.

Wird die Funktion mit einem Parameter aufgerufen, wird der Wert dieser Einstellung zurückgegeben. Beim Aufruf mit zwei Parametern wird die Einstellung auf den angegebenen Wert geändert.

readline_list_history (PHP 4 >= 4.0.0)

Auflistung der History

array **readline_list_history** (void)

Diese Funktion gibt ein Array der gesamten Kommandozeilen-History zurück. Die Elemente sind numerisch(integer) indiziert, beginnend bei Null.

readline_read_history (PHP 4 >= 4.0.0)

Liest die History

bool **readline_read_history** (string dateiname)

Diese Funktion liest eine Kommando-History aus einer Datei.

readline_write_history (PHP 4 >= 4.0.0)

Schreibt die History

bool **readline_write_history** (string dateiname)

Diese Funktion schreibt die Kommando-History in eine Datei.

LXXXIV. GNU Recode Funktionen

Dieses Modul stellt eine Schnittstelle zu der GNU Recode Bibliothek Version 3.5 dar. Um die Umkodierungsfunktionen benutzen zu können muss die GNU Recode Bibliothek ab Version 3.5 auf dem System installiert und PHP mit der Option --with-recode kompiliert werden.

Die GNU Recode Bibliothek ermöglicht die Konversion zwischen verschiedenen Zeichensätzen (wie ASCII, ISO Latin-1, IBM-PC und weiteren) und Eingabekonventionen (z.B. HTML oder LaTeX). Wenn eine exakte Umkodierung nicht möglich ist, werden die störenden Zeichen entfernt oder durch Annäherungen ersetzt. Die Bibliothek erkennt und verwendet etwa 150 verschiedene Zeichensätze und kann Umwandlungen zwischen den meisten dieser Zeichensätze vornehmen. Fast alle der in RFC 1345 beschriebenen Zeichensätze werden unterstützt.

recode_string (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Umkodierung eines Strings entsprechend einer Recode Anweisung

string **recode_string** (string request, string string)

Der Parameter *string* wird entsprechend der Recode Anweisung *request* umgewandelt. Zurückgegeben wird der umkodierte String oder FALSE, wenn eine Umkodierung nicht möglich ist.

Eine einfache Recode Anweisung wäre beispielsweise "lat1..iso646-de". Detaillierte Beschreibungen zu Recode Anweisungen sind in der GNU Recode Dokumentation auf Ihrem System zu finden.

Beispiel 1. Einfaches recode_string() Beispiel:

```
print recode_string ("us..flat", "Der folgende Buchstabe hat ein diakritisches Zei-
chen: &aacute;");
```

recode (PHP 4 >= 4.0.0)

Umkodierung eines Strings entsprechend einer Recode Anweisung

string **recode** (string request, string string)

Anmerkung: Dies ist ein Alias für recode_string(), der in PHP 4 hinzugefügt wurde.

recode_file (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Umkodierung von Dateien entsprechend der Recode Anweisung

bool **recode_file** (string request, resource input, resource output)

Umkodierung der Datei, die durch den Dateizeiger *input* referenziert wird in die Datei *output*, entsprechend der Recode Anweisung *request*. Gibt TRUE bei erfolgter Umwandlung, sonst FALSE zurück.

Diese Funktion arbeitet momentan nicht mit externen Dateien (URLs). Die beiden Dateizeiger müssen auf lokale Dateien verweisen.

Beispiel 1. Einfaches recode_file() Beispiel

```
$input = fopen ('input.txt', 'r');
$output = fopen ('output.txt', 'w');
recode_file ("us..flat", $input, $output);
```

LXXXV. Regular Expression Functions (Perl-Compatible)

> **Anmerkung:** PHP also supports regular expressions using a POSIX-extended syntax using the POSIX-extended regex functions..

The syntax for patterns used in these functions closely resembles Perl. The expression should be enclosed in the delimiters, a forward slash (/), for example. Any character can be used for delimiter as long as it's not alphanumeric or backslash (\). If the delimiter character has to be used in the expression itself, it needs to be escaped by backslash. Since PHP 4.0.4, you can also use Perl-style (), {}, [], and <> matching delimiters.

The ending delimiter may be followed by various modifiers that affect the matching. See Pattern Modifiers.

Beispiel 1. Examples of valid patterns

- /<\/\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/
- {^\s+(\s+)?$}

Beispiel 2. Examples of invalid patterns

- /href='(.*)' - missing ending delimiter
- /\w+\s*\w+/J - unknown modifier 'J'
- 1-\d3-\d3-\d4| - missing starting delimiter

> **Anmerkung:** The Perl-compatible regular expression functions are available in PHP 4 and in PHP 3.0.9 and up.
>
> Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. It is available at ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

preg_match (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Perform a regular expression match

int **preg_match** (string pattern, string subject [, array matches])

Searches *subject* for a match to the regular expression given in *pattern*.

If *matches* is provided, then it is filled with the results of search. $matches[0] will contain the text that matched the full pattern, $matches[1] will have the text that matched the first captured parenthesized subpattern, and so on.

Returns TRUE if a match for *pattern* was found in the subject string, or FALSE if not match was found or an error occurred.

Beispiel 1. Find the string of text "php"

```
// the "i" after the pattern delimiter indicates a case-insensitive search
if (preg_match ("/php/i", "PHP is the web scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

Beispiel 2. find the word "web"

```
// the \b in the pattern indicates a word boundary, so only the distinct
// word "web" is matched, and not a word partial like "webbing" or "cobweb"
if (preg_match ("/\bweb\b/i", "PHP is the web scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
if (preg_match ("/\bweb\b/i", "PHP is the website scripting language of choi-
ce.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

Beispiel 3. Getting the domain name out of a URL

```
// get host name from URL
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.php.net/index.html", $matches);
$host = $matches[2];
// get last two segments of host name
preg_match("/[^\.\/]+\.[^\.\/]+$/",$host,$matches);
```

```
echo "domain name is: ".$matches[0]."\n";
```

This example will produce:

```
domain name is: php.net
```

See also preg_match_all(), preg_replace(), and preg_split().

preg_match_all (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Perform a global regular expression match

int **preg_match_all** (string pattern, string subject, array matches [, int order])

Searches *subject* for all matches to the regular expression given in *pattern* and puts them in *matches* in the order specified by *order*.

After the first match is found, the subsequent searches are continued on from end of the last match.

order can be one of two things:

PREG_PATTERN_ORDER

Orders results so that $matches[0] is an array of full pattern matches, $matches[1] is an array of strings matched by the first parenthesized subpattern, and so on.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_PATTERN_ORDER);
print $out[0][0]., ".$out[0][1]."\n";
print $out[1][0]., ".$out[1][1]."\n";
```

This example will produce:

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

So, $out[0] contains array of strings that matched full pattern, and $out[1] contains array of strings enclosed by tags.

PREG_SET_ORDER

Orders results so that $matches[0] is an array of first set of matches, $matches[1] is an array of second set of matches, and so on.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_SET_ORDER);
print $out[0][0]., ".$out[0][1]."\n";
print $out[1][0]., ".$out[1][1]."\n";
```

This example will produce:

```
<b>example: </b>, example:
<div align=left>this is a test</div>, this is a test
```

In this case, $matches[0] is the first set of matches, and $matches[0][0] has text matched by full pattern, $matches[0][1] has text matched by first subpattern and so on. Similarly, $matches[1] is the second set of matches, etc.

If `order` is not specified, it is assumed to be PREG_PATTERN_ORDER.

Returns the number of full pattern matches, or `FALSE` if no match is found or an error occurred.

Beispiel 1. Getting all phone numbers out of some text.

```
preg_match_all ("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
                "Call 555-1212 or 1-800-555-1212", $phones);
```

Beispiel 2. Find matching HTML tags (greedy)

```
// The \\2 is an example of backreferencing. This tells pcre that
// it must match the second set of parentheses in the regular expression
// itself, which would be the ([\w]+) in this case. The extra backslash is
// required because the string is in double quotes.
$html = "<b>bold text</b><a href=howdy.html>click me</a>;

preg_match_all ("/(<([\w]+)[^>]*>)(.*)(<\/\\2>)/", $html, $matches);

for ($i=0; $i< count($matches[0]); $i++) {
  echo "matched: ".$matches[0][$i]."\n";
  echo "part 1: ".$matches[1][$i]."\n";
  echo "part 2: ".$matches[3][$i]."\n";
```

```
  echo "part 3: ".$matches[4][$i]."\n\n";
}
```

This example will produce:

```
matched: <b>bold text</b>
part 1: <b>
part 2: bold text
part 3: </b>

matched: <a href=howdy.html>click me</a>
part 1: <a href=howdy.html>
part 2: click me
part 3: </a>
```

See also preg_match(), preg_replace(), and preg_split().

preg_replace (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Perform a regular expression search and replace

mixed **preg_replace** (mixed pattern, mixed replacement, mixed subject [, int limit])

Searches *subject* for matches to *pattern* and replaces them with *replacement*. If *limit* is specified, then only *limit* matches will be replaced; if *limit* is omitted or is -1, then all matches are replaced.

Replacement may contain references of the form *n* or (since PHP 4.0.4) *$n*, with the latter form being the preferred one. Every such reference will be replaced by the text captured by the *n*'th parenthesized pattern. *n* can be from 0 to 99, and \\0 or $0 refers to the text matched by the whole pattern. Opening parentheses are counted from left to right (starting from 1) to obtain the number of the capturing subpattern.

If matches are found, the new *subject* will be returned, otherwise *subject* will be returned unchanged.

Every parameter to **preg_replace()** can be an array.

If *subject* is an array, then the search and replace is performed on every entry of *subject*, and the return value is an array as well.

If *pattern* and *replacement* are arrays, then **preg_replace()** takes a value from each array and uses them to do search and replace on *subject*. If *replacement* has fewer values than *pattern*, then empty string is used for the rest of replacement values. If *pattern* is an array and *replacement* is a string, then this replacement string is used for every value of *pattern*. The converse would not make sense, though.

/e modifier makes **preg_replace()** treat the *replacement* parameter as PHP code after the appropriate references substitution is done. Tip: make sure that *replacement* constitutes a valid

PHP code string, otherwise PHP will complain about a parse error at the line containing **preg_replace()**.

Beispiel 1. Replacing several values

```
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                   "/^\s*{(\w+)}\s*=/");
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($patterns, $replace, "{startDate} = 1999-5-27");
```

This example will produce:

```
$startDate = 5/27/1999
```

Beispiel 2. Using /e modifier

```
preg_replace ("/(<\/?)(\w+)([^>]*>)/e",
              "'\\1'.strtoupper('\\2').'\\3'",
              $html_body);
```

This would capitalize all HTML tags in the input text.

Beispiel 3. Convert HTML to text

```
// $document should contain an HTML document.
// This will remove HTML tags, javascript sections
// and white space. It will also convert some
// common HTML entities to their text equivalent.

$search = array ("'<script[^>]*?>.*?</script>'si",  // Strip out javascript
                 "'<[\/\!]*?[^<>]*?>'si",           // Strip out html tags
                 "'([\r\n])[\s]+'",                 // Strip out white space
                 "'&(quot|#34);'i",                 // Replace html entities
                 "'&(amp|#38);'i",
                 "'&(lt|#60);'i",
                 "'&(gt|#62);'i",
                 "'&(nbsp|#160);'i",
                 "'&(iexcl|#161);'i",
                 "'&(cent|#162);'i",
                 "'&(pound|#163);'i",
                 "'&(copy|#169);'i",
                 "'&#(\d+);'e");                    // evaluate as php

$replace = array ("",
```

```
                    "",
                    "\\1",
                    "\"",
                    "&",
                    "<",
                    ">",
                    " ",
                    chr(161),
                    chr(162),
                    chr(163),
                    chr(169),
                    "chr(\\1)");

$text = preg_replace ($search, $replace, $document);
```

Anmerkung: Parameter *limit* was added after PHP 4.0.1pl2.

See also preg_match(), preg_match_all(), and preg_split().

preg_replace_callback (PHP 4 >= 4.0.5)

Perform a regular expression search and replace using a callback

mixed **preg_replace_callback** (mixed pattern, mixed callback, mixed subject [, int limit])

The behavior of this function is almost identical to preg_replace(), except for the fact that instead of *replacement* parameter, one should specify a *callback* that will be called and passed an array of matched elements in the subject string. The callback should return the replacement string. This function was added in PHP 4.0.5.

See also preg_replace().

preg_split (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Split string by a regular expression

array **preg_split** (string pattern, string subject [, int limit [, int flags]])

Anmerkung: Parameter *flags* was added in PHP 4 Beta 3.

Returns an array containing substrings of *subject* split along boundaries matched by *pattern*.

If *limit* is specified, then only substrings up to *limit* are returned, and if *limit* is -1, it actually means "no limit", which is useful for specifying the *flags*.

flags can be any combination of the following flags (combined with bitwise | operator):

PREG_SPLIT_NO_EMPTY

If this flag is set, only non-empty pieces will be returned by **preg_split()**.

PREG_SPLIT_DELIM_CAPTURE

If this flag is set, parenthesized expression in the delimiter pattern will be captured and returned as well. This flag was added for 4.0.5.

Beispiel 1. preg_split() example : Get the parts of a search string.

```
// split the phrase by any number of commas or space characters,
// which include " ", \r, \t, \n and \f
$keywords = preg_split ("/[\s,]+/", "hypertext language, programming");
```

Beispiel 2. Splitting a string into component characters.

```
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
```

See also spliti(), split(), implode(), preg_match(), preg_match_all(), and preg_replace().

preg_quote (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Quote regular expression characters

string **preg_quote** (string str [, string delimiter])

preg_quote() takes *str* and puts a backslash in front of every character that is part of the regular expression syntax. This is useful if you have a run-time string that you need to match in some text and the string may contain special regex characters.

If the optional *delimiter* is specified, it will also be escaped. This is useful for escaping the delimiter that is required by the PCRE functions. The / is the most commonly used delimiter.

The special regular expression characters are:

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Beispiel 1.

```
$keywords = "$40 for a g3/400";
$keywords = preg_quote ($keywords, "/");
echo $keywords; // returns \$40 for a g3\/400
```

Beispiel 2. Italicizing a word within some text

```
// In this example, preg_quote($word) is used to keep the
// asterisks from having special meaning to the regular
// expression.

$textbody = "This book is *very* difficult to find.";
$word = "*very*";
$textbody = preg_replace ("/".preg_quote($word)."/",
                          "<i>".$word."</i>",
                          $textbody);
```

preg_grep (PHP 4 >= 4.0.0)

Return array entries that match the pattern

array **preg_grep** (string pattern, array input)

preg_grep() returns the array consisting of the elements of the *input* array that match the given *pattern*.

Since PHP 4.0.4, the results returned by **preg_grep()** are indexed using the keys from the input array. If this behavior is undesirable, use array_values() on the array returned by **preg_grep()** to reindex the values.

Beispiel 1. preg_grep() example

```
// return all array elements
// containing floating point numbers
$fl_array = preg_grep ("/^(\d+)?\.\d+$/", $array);
```

Pattern Modifiers (unknown)

Describes possible modifiers in regex patterns

The current possible PCRE modifiers are listed below. The names in parentheses refer to internal PCRE names for these modifiers.

i (PCRE_CASELESS)

If this modifier is set, letters in the pattern match both upper and lower case letters.

m (PCRE_MULTILINE)

By default, PCRE treats the subject string as consisting of a single "line" of characters (even if it actually contains several newlines). The "start of line" metacharacter (^) matches only at the start of the string, while the "end of line" metacharacter ($) matches only at the end of the string, or before a terminating newline (unless *D* modifier is set). This is the same as Perl.

When this modifier is set, the "start of line" and "end of line" constructs match immediately following or immediately before any newline in the subject string, respectively, as well as at the very start and end. This is equivalent to Perl's /m modifier. If there are no "\n" characters in a subject string, or no occurrences of ^ or $ in a pattern, setting this modifier has no effect.

s (PCRE_DOTALL)

If this modifier is set, a dot metacharacter in the pattern matches all characters, including newlines. Without it, newlines are excluded. This modifier is equivalent to Perl's /s modifier. A negative class such as [^a] always matches a newline character, independent of the setting of this modifier.

x (PCRE_EXTENDED)

If this modifier is set, whitespace data characters in the pattern are totally ignored except when escaped or inside a character class, and characters between an unescaped # outside a character class and the next newline character, inclusive, are also ignored. This is equivalent to Perl's /x modifier, and makes it possible to include comments inside complicated patterns. Note, however, that this applies only to data characters. Whitespace characters may never appear within special character sequences in a pattern, for example within the sequence (?(which introduces a conditional subpattern.

e

If this modifier is set, preg_replace() does normal substitution of backreferences in the replacement string, evaluates it as PHP code, and uses the result for replacing the search string.

Only preg_replace() uses this modifier; it is ignored by other PCRE functions.

A (PCRE_ANCHORED)

If this modifier is set, the pattern is forced to be "anchored", that is, it is constrained to match only at the start of the string which is being searched (the "subject string"). This effect can also be achieved by appropriate constructs in the pattern itself, which is the only way to do it in Perl.

D (PCRE_DOLLAR_ENDONLY)

If this modifier is set, a dollar metacharacter in the pattern matches only at the end of the subject string. Without this modifier, a dollar also matches immediately before the final character if it is a newline (but not before any other newlines). This modifier is ignored if *m* modifier is set. There is no equivalent to this modifier in Perl.

S

When a pattern is going to be used several times, it is worth spending more time analyzing it in order to speed up the time taken for matching. If this modifier is set, then this extra analysis is performed. At present, studying a pattern is useful only for non-anchored patterns that do not have a single fixed starting character.

U (PCRE_UNGREEDY)

This modifier inverts the "greediness" of the quantifiers so that they are not greedy by default, but become greedy if followed by "?". It is not compatible with Perl. It can also be set by a (?U) modifier setting within the pattern.

X (PCRE_EXTRA)

This modifier turns on additional functionality of PCRE that is incompatible with Perl. Any backslash in a pattern that is followed by a letter that has no special meaning causes an error, thus reserving these combinations for future expansion. By default, as in Perl, a backslash followed by a letter with no special meaning is treated as a literal. There are at present no other features controlled by this modifier.

u (PCRE_UTF8)

This modifier turns on additional functionality of PCRE that is incompatible with Perl. Pattern strings are treated as UTF-8. This modifier is available from PHP 4.1.0 or greater.

Pattern Syntax (unknown)

Describes PCRE regex syntax

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5, with just a few differences (see below). The current implementation corresponds to Perl 5.005.

The differences described here are with respect to Perl 5.005.

1. By default, a whitespace character is any character that the C library function isspace() recognizes, though it is possible to compile PCRE with alternative character type tables. Normally isspace() matches space, formfeed, newline, carriage return, horizontal tab, and vertical tab. Perl 5 no longer includes vertical tab in its set of whitespace characters. The \v escape that was in the Perl documentation for a long time was never in fact recognized. However, the character itself was treated as whitespace at least up to 5.002. In 5.004 and 5.005 it does not match \s.
2. PCRE does not allow repeat quantifiers on lookahead assertions. Perl permits them, but they do not mean what you might think. For example, (?!a){3} does not assert that the next three characters are not "a". It just asserts that the next character is not "a" three times.
3. Capturing subpatterns that occur inside negative looka- head assertions are counted, but their entries in the offsets vector are never set. Perl sets its numerical vari- ables from any such patterns that are matched before the assertion fails to match something (thereby succeeding), but only if the negative lookahead assertion contains just one branch.

4. Though binary zero characters are supported in the sub- ject string, they are not allowed in a pattern string because it is passed as a normal C string, terminated by zero. The escape sequence "\0" can be used in the pattern to represent a binary zero.
5. The following Perl escape sequences are not supported: \l, \u, \L, \U, \E, \Q. In fact these are implemented by Perl's general string-handling and are not part of its pat- tern matching engine.
6. The Perl \G assertion is not supported as it is not relevant to single pattern matches.
7. Fairly obviously, PCRE does not support the (?{code}) construction.
8. There are at the time of writing some oddities in Perl 5.005_02 concerned with the settings of captured strings when part of a pattern is repeated. For example, matching "aba" against the pattern /^(a(b)?)+$/ sets $2 to the value "b", but matching "aabbaa" against /^(aa(bb)?)+$/ leaves $2 unset. However, if the pattern is changed to /^(aa(b(b))?)+$/ then $2 (and $3) get set. In Perl 5.004 $2 is set in both cases, and that is also `TRUE` of PCRE. If in the future Perl changes to a consistent state that is different, PCRE may change to follow.
9. Another as yet unresolved discrepancy is that in Perl 5.005_02 the pattern /^(a)?(?(1)a|b)+$/ matches the string "a", whereas in PCRE it does not. However, in both Perl and PCRE /^(a)?a/ matched against "a" leaves $1 unset.
10. PCRE provides some extensions to the Perl regular expression facilities:
    a. Although lookbehind assertions must match fixed length strings, each alternative branch of a lookbehind assertion can match a different length of string. Perl 5.005 requires them all to have the same length.
    b. If PCRE_DOLLAR_ENDONLY is set and PCRE_MULTILINE is not set, the $ meta-character matches only at the very end of the string.
    c. If PCRE_EXTRA is set, a backslash followed by a letter with no special meaning is faulted.
    d. If PCRE_UNGREEDY is set, the greediness of the repeti- tion quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

Introduction

The syntax and semantics of the regular expressions sup- ported by PCRE are described below. Regular expressions are also described in the Perl documentation and in a number of other books, some of which have copious examples. Jeffrey Friedl's "Mastering Regular Expressions", published by O'Reilly (ISBN 1-56592-257-3), covers them in great detail. The description here is intended as reference documentation. A regular expression is a pattern that is matched against a subject string from left to right. Most characters stand for themselves in a pattern, and match the corresponding charac- ters in the subject. As a trivial example, the pattern `The quick brown fox` matches a portion of a subject string that is identical to itself.

Meta-caracters

The power of regular expressions comes from the ability to include alternatives and repetitions in the pat- tern. These are encoded in the pattern by the use of *meta- characters*, which do not stand for themselves but instead are interpreted in some special way.

There are two different sets of meta-characters: those that are recognized anywhere in the pattern except within square brackets, and those that are recognized in square brackets. Outside square brackets, the meta-characters are as follows:

\

general escape character with several uses

^

assert start of subject (or line, in multiline mode)

$

assert end of subject (or line, in multiline mode)

.

match any character except newline (by default)

[

start character class definition

]

end character class definition

|

start of alternative branch

(

start subpattern

)

end subpattern

?

extends the meaning of (, also 0 or 1 quantifier, also quantifier minimizer

*

0 or more quantifier

+

1 or more quantifier

{

start min/max quantifier

}

end min/max quantifier

Part of a pattern that is in square brackets is called a "character class". In a character class the only meta- characters are:

\

general escape character

^

negate the class, but only if the first character

-

indicates character range

]

terminates the character class

The following sections describe the use of each of the meta-characters.

backslash

The backslash character has several uses. Firstly, if it is followed by a non-alphameric character, it takes away any special meaning that character may have. This use of backslash as an escape character applies both inside and outside character classes.

For example, if you want to match a "*" character, you write "*" in the pattern. This applies whether or not the follow- ing character would otherwise be interpreted as a meta- character, so it is always safe to precede a non-alphameric with "\" to specify that it stands for itself. In particu- lar, if you want to match a backslash, you write "\\".

If a pattern is compiled with the PCRE_EXTENDED option, whi- tespace in the pattern (other than in a character class) and characters between a "#" outside a character class and the next newline character are ignored. An escaping backslash can be used to include a whitespace or "#" character as part of the pattern.

A second use of backslash provides a way of encoding non- printing characters in patterns in a visible manner. There is no restriction on the appearance of non-printing charac- ters, apart from the binary zero that terminates a pattern, but when a pattern is being prepared by text editing, it is usually easier to use one of the following escape sequences than the binary character it represents:

\a

alarm, that is, the BEL character (hex 07)

\cx

"control-x", where x is any character

\e

escape (hex 1B)

\f

formfeed (hex 0C)

\n

newline (hex 0A)

\r

carriage return (hex 0D)

\t

tab (hex 09)

\xhh

character with hex code hh

\ddd

character with octal code ddd, or backreference

The precise effect of "`\cx`" is as follows: if "`x`" is a lower case letter, it is converted to upper case. Then bit 6 of the character (hex 40) is inverted. Thus "`\cz`" becomes hex 1A, but "`\c{`" becomes hex 3B, while "`\c;`" becomes hex 7B.

After "`\x`", up to two hexadecimal digits are read (letters can be in upper or lower case).

After "`\0`" up to two further octal digits are read. In both cases, if there are fewer than two digits, just those that are present are used. Thus the sequence "`\0\x\07`" specifies two binary zeros followed by a BEL character. Make sure you supply two digits after the initial zero if the character that follows is itself an octal digit.

The handling of a backslash followed by a digit other than 0 is complicated. Outside a character class, PCRE reads it and any following digits as a decimal number. If the number is less than 10, or if there have been at least that many previous capturing left parentheses in the expression, the entire sequence is taken as a *back reference*. A description of how this works is given later, following the discussion of parenthesized subpatterns.

Inside a character class, or if the decimal number is greater than 9 and there have not been that many capturing subpatterns, PCRE re-reads up to three octal digits follow- ing the backslash, and generates a single byte from the least significant 8 bits of the value. Any subsequent digits stand for themselves. For example:

\040

is another way of writing a space

\40

is the same, provided there are fewer than 40 previous capturing subpatterns

\7

is always a back reference

\11

might be a back reference, or another way of writing a tab

\011

is always a tab

\0113

is a tab followed by the character "3"

\113

is the character with octal code 113 (since there can be no more than 99 back references)

\377

is a byte consisting entirely of 1 bits

\81

is either a back reference, or a binary zero followed by the two characters "8" and "1"

Note that octal values of 100 or greater must not be intro- duced by a leading zero, because no more than three octal digits are ever read.

All the sequences that define a single byte value can be used both inside and outside character classes. In addition, inside a character class, the sequence "`\b`" is interpreted as the backspace character (hex 08). Outside a character class it has a different meaning (see below).

The third use of backslash is for specifying generic charac- ter types:

\d

any decimal digit

\D

any character that is not a decimal digit

\s

any whitespace character

\S

any character that is not a whitespace character

\w

any "word" character

\W

any "non-word" character

Each pair of escape sequences partitions the complete set of characters into two disjoint sets. Any given character matches one, and only one, of each pair.

A "word" character is any letter or digit or the underscore character, that is, any character which can be part of a Perl "`word`". The definition of letters and digits is controlled by PCRE's character tables, and may vary if locale-specific matching is taking place (see "Locale support" above). For example, in the "fr" (French) locale, some char- acter codes greater than 128 are used for accented letters, and these are matched by `\w`.

These character type sequences can appear both inside and outside character classes. They each match one character of the appropriate type. If the current matching point is at the end of the subject string, all of them fail, since there is no character to match.

The fourth use of backslash is for certain simple asser- tions. An assertion specifies a condition that has to be met at a particular point in a match, without consuming any characters from the subject

string. The use of subpatterns for more complicated assertions is described below. The backslashed assertions are

\b

word boundary

\B

not a word boundary

\A

start of subject (independent of multiline mode)

\Z

end of subject or newline at end (independent of multiline mode)

\z

end of subject (independent of multiline mode)

These assertions may not appear in character classes (but note that "`\b`" has a different meaning, namely the backspace character, inside a character class).

A word boundary is a position in the subject string where the current character and the previous character do not both match `\w` or `\W` (i.e. one matches `\w` and the other matches `\W`), or the start or end of the string if the first or last character matches \w, respectively.

The `\A`, `\Z`, and `\z` assertions differ from the traditional circumflex and dollar (described below) in that they only ever match at the very start and end of the subject string, whatever options are set. They are not affected by the PCRE_NOTBOL or PCRE_NOTEOL options. The difference between `\Z` and `\z` is that `\Z` matches before a newline that is the last character of the string as well as at the end of the string, whereas `\z` matches only at the end.

Circumflex and dollar

Outside a character class, in the default matching mode, the circumflex character is an assertion which is true only if the current matching point is at the start of the subject string. Inside a character class, circumflex has an entirely different meaning (see below).

Circumflex need not be the first character of the pattern if a number of alternatives are involved, but it should be the first thing in each alternative in which it appears if the pattern is ever to match that branch. If all possible alternatives start with a circumflex, that is, if the pattern is constrained to match only at the start of the subject, it is said to be an "anchored" pattern. (There are also other constructs that can cause a pattern to be anchored.)

A dollar character is an assertion which is `TRUE` only if the current matching point is at the end of the subject string,

or immediately before a newline character that is the last character in the string (by default). Dollar need not be the last character of the pattern if a number of alternatives are involved, but it should be the last item in any branch in which it appears. Dollar has no special meaning in a character class.

The meaning of dollar can be changed so that it matches only at the very end of the string, by setting the PCRE_DOLLAR_ENDONLY option at compile or matching time. This does not affect the \Z assertion.

The meanings of the circumflex and dollar characters are changed if the PCRE_MULTILINE option is set. When this is the case, they match immediately after and immediately before an internal "\n" character, respectively, in addition to matching at the start and end of the subject string. For example, the pattern /^abc$/ matches the subject string "def\nabc" in multiline mode, but not otherwise. Consequently, patterns that are anchored in single line mode because all branches start with "^" are not anchored in multiline mode. The PCRE_DOLLAR_ENDONLY option is ignored if PCRE_MULTILINE is set.

Note that the sequences \A, \Z, and \z can be used to match the start and end of the subject in both modes, and if all branches of a pattern start with \A is it always anchored, whether PCRE_MULTILINE is set or not.

FULL STOP

Outside a character class, a dot in the pattern matches any one character in the subject, including a non-printing character, but not (by default) newline. If the PCRE_DOTALL option is set, then dots match newlines as well. The handling of dot is entirely independent of the handling of circumflex and dollar, the only relationship being that they both involve newline characters. Dot has no special meaning in a character class.

Square brackets

An opening square bracket introduces a character class, terminated by a closing square bracket. A closing square bracket on its own is not special. If a closing square bracket is required as a member of the class, it should be the first data character in the class (after an initial cir-

cumflex, if present) or escaped with a backslash.

A character class matches a single character in the subject; the character must be in the set of characters defined by the class, unless the first character in the class is a circumflex, in which case the subject character must not be in the set defined by the class. If a circumflex is actually required as a member of the class, ensure it is not the first character, or escape it with a backslash.

For example, the character class [aeiou] matches any lower case vowel, while [^aeiou] matches any character that is not a lower case vowel. Note that a circumflex is just a convenient notation for specifying the characters which are in the class by enumerating those that are not. It is not an assertion: it still consumes a character from the subject string, and fails if the current pointer is at the end of the string.

When caseless matching is set, any letters in a class represent both their upper case and lower case versions, so for example, a caseless [aeiou] matches "A" as well as "a", and a caseless [^aeiou] does not match "A", whereas a caseful version would.

The newline character is never treated in any special way in character classes, whatever the setting of the PCRE_DOTALL or PCRE_MULTILINE options is. A class such as [^a] will always match a newline.

The minus (hyphen) character can be used to specify a range of characters in a character class. For example, [d-m] matches any letter between d and m, inclusive. If a minus character is required in a class, it must be escaped with a backslash or appear in a position where it cannot be interpreted as indicating a range, typically as the first or last character in the class.

It is not possible to have the literal character "]" as the end character of a range. A pattern such as [W-]46] is interpreted as a class of two characters ("W" and "-") followed by a literal string "46]", so it would match "W46]" or "-46]". However, if the "]" is escaped with a backslash it is interpreted as the end of range, so [W-\]46] is interpreted as a single class containing a range followed by two separate characters. The octal or hexadecimal representation of "]" can also be used to end a range.

Ranges operate in ASCII collating sequence. They can also be used for characters specified numerically, for example [\000-\037]. If a range that includes letters is used when caseless matching is set, it matches the letters in either

case. For example, [W-c] is equivalent to []\[\^_‘wxyzabc], matched caselessly, and if character tables for the "fr" locale are in use, [\xc8-\xcb] matches accented E characters in both cases.

The character types \d, \D, \s, \S, \w, and \W may also appear in a character class, and add the characters that they match to the class. For example, [\dABCDEF] matches any hexadecimal digit. A circumflex can conveniently be used with the upper case character types to specify a more restricted set of characters than the matching lower case type. For example, the class [^\W_] matches any letter or digit, but not underscore.

All non-alphameric characters other than \, -, ^ (at the start) and the terminating] are non-special in character classes, but it does no harm if they are escaped.

Vertical bar

Vertical bar characters are used to separate alternative patterns. For example, the pattern

```
gilbert|sullivan
```

matches either "gilbert" or "sullivan". Any number of alternatives may appear, and an empty alternative is permitted (matching the empty string). The matching process tries each alternative in turn, from left to right, and the first one that succeeds is used. If the alternatives are within a subpattern (defined below), "succeeds" means matching the rest of the main pattern as well as the alternative in the subpattern.

Internal option setting

The settings of PCRE_CASELESS ,
PCRE_MULTILINE ,
PCRE_DOTALL ,
and PCRE_EXTENDED can be changed from within the pattern by a sequence of Perl option letters enclosed between "(?" and ")". The option letters are

```
i  for PCRE_CASELESS
m  for PCRE_MULTILINE
s  for PCRE_DOTALL
x  for PCRE_EXTENDED
```

For example, (?im) sets caseless, multiline matching. It is also possible to unset these options by preceding the letter with a hyphen, and a combined setting and unsetting such as (?im-sx), which sets PCRE_CASELESS and PCRE_MULTILINE while unsetting PCRE_DOTALL and PCRE_EXTENDED , is also permitted. If a letter appears both before and after the hyphen, the option is unset.

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern (defined below), the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

which in turn is the same as compiling the pattern abc with PCRE_CASELESS set. In other words, such "top level" settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at top level, the rightmost setting is used.

If an option change occurs inside a subpattern, the effect is different. This is a change of behaviour in Perl 5.005. An option change inside a subpattern affects only that part of the subpattern that follows it, so

```
(a(?i)b)c
```

matches abc and aBc and no other strings (assuming PCRE_CASELESS is not used). By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example,

```
(a(?i)b|c)
```

matches "ab", "aB", "c", and "C", even though when matching "C" the first branch is abandoned before the option setting. This is because the effects of option settings happen at compile time. There would be some very weird behaviour otherwise.

The PCRE-specific options PCRE_UNGREEDY and PCRE_EXTRA can
be changed in the same way as the Perl-compatible options by

using the characters U and X respectively. The (?X) flag setting is special in that it must always occur earlier in the pattern than any of the additional features it turns on, even when it is at top level. It is best put at the start.

subpatterns

Subpatterns are delimited by parentheses (round brackets), which can be nested. Marking part of a pattern as a subpattern does two things:

1. It localizes a set of alternatives. For example, the pattern

```
cat(aract|erpillar|)
```

matches one of the words "cat", "cataract", or "caterpillar". Without the parentheses, it would match "cataract", "erpillar" or the empty string.

2. It sets up the subpattern as a capturing subpattern (as defined above). When the whole pattern matches, that portion of the subject string that matched the subpattern is passed back to the caller via the *ovector* argument of **pcre_exec()**. Opening parentheses are counted from left to right (starting from 1) to obtain the numbers of the capturing subpatterns.

For example, if the string "the red king" is matched against the pattern

```
the ((red|white) (king|queen))
```

the captured substrings are "red king", "red", and "king", and are numbered 1, 2, and 3.

The fact that plain parentheses fulfil two functions is not always helpful. There are often times when a grouping subpattern is required without a capturing requirement. If an opening parenthesis is followed by "?:", the subpattern does not do any capturing, and is not counted when computing the number of any subsequent capturing subpatterns. For example, if the string "the white queen" is matched against the pattern

```
the ((?:red|white) (king|queen))
```

the captured substrings are "white queen" and "queen", and are numbered 1 and 2. The maximum number of captured sub-

strings is 99, and the maximum number of all subpatterns, both capturing and non-capturing, is 200.

As a convenient shorthand, if any option settings are required at the start of a non-capturing subpattern, the option letters may appear between the "?" and the ":". Thus the two patterns

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

match exactly the same set of strings. Because alternative branches are tried from left to right, and options are not reset until the end of the subpattern is reached, an option setting in one branch does affect subsequent branches, so the above patterns match "SUNDAY" as well as "Saturday".

Repetition

Repetition is specified by quantifiers, which can follow any of the following items:

a single character, possibly escaped
the . metacharacter
a character class
a back reference (see next section)
a parenthesized subpattern (unless it is an assertion - see below)

The general repetition quantifier specifies a minimum and maximum number of permitted matches, by giving the two numbers in curly brackets (braces), separated by a comma. The numbers must be less than 65536, and the first must be less than or equal to the second. For example:

```
z{2,4}
```

matches "zz", "zzz", or "zzzz". A closing brace on its own is not a special character. If the second number is omitted, but the comma is present, there is no upper limit; if the second number and the comma are both omitted, the quantifier specifies an exact number of required matches. Thus

```
[aeiou]{3,}
```

matches at least 3 successive vowels, but may match many more, while

```
\d{8}
```

matches exactly 8 digits. An opening curly bracket that appears in a position where a quantifier is not allowed, or one that does not match the syntax of a quantifier, is taken as a literal character. For example, {,6} is not a quantifier, but a literal string of four characters.

The quantifier {0} is permitted, causing the expression to behave as if the previous item and the quantifier were not present.

For convenience (and historical compatibility) the three most common quantifiers have single-character abbreviations:

```
*    is equivalent to {0,}
+    is equivalent to {1,}
?    is equivalent to {0,1}
```

It is possible to construct infinite loops by following a subpattern that can match no characters with a quantifier that has no upper limit, for example:

```
(a?)*
```

Earlier versions of Perl and PCRE used to give an error at compile time for such patterns. However, because there are cases where this can be useful, such patterns are now accepted, but if any repetition of the subpattern does in fact match no characters, the loop is forcibly broken.

By default, the quantifiers are "greedy", that is, they match as much as possible (up to the maximum number of permitted times), without causing the rest of the pattern to fail. The classic example of where this gives problems is in trying to match comments in C programs. These appear between the sequences /* and */ and within the sequence, individual * and / characters may appear. An attempt to match C comments by applying the pattern

```
/\*.*\*/
```

to the string

```
/* first command */  not comment  /* second comment */
```

fails, because it matches the entire string due to the greediness of the .* item.

However, if a quantifier is followed by a question mark, then it ceases to be greedy, and instead matches the minimum number of times possible, so the pattern

```
/\*.*?\*/
```

does the right thing with the C comments. The meaning of the various quantifiers is not otherwise changed, just the preferred number of matches. Do not confuse this use of question mark with its use as a quantifier in its own right. Because it has two uses, it can sometimes appear doubled, as in

```
\d??\d
```

which matches one digit by preference, but can match two if that is the only way the rest of the pattern matches.

If the PCRE_UNGREEDY option is set (an option which is not available in Perl) then the quantifiers are not greedy by default, but individual ones can be made greedy by following them with a question mark. In other words, it inverts the default behaviour.

When a parenthesized subpattern is quantified with a minimum repeat count that is greater than 1 or with a limited maximum, more store is required for the compiled pattern, in proportion to the size of the minimum or maximum.

If a pattern starts with .* or .{0,} and the PCRE_DOTALL option (equivalent to Perl's /s) is set, thus allowing the . to match newlines, then the pattern is implicitly anchored, because whatever follows will be tried against every character position in the subject string, so there is no point in retrying the overall match at any position after the first. PCRE treats such a pattern as though it were preceded by \A. In cases where it is known that the subject string contains no newlines, it is worth setting PCRE_DOTALL when the pattern begins with .* in order to obtain this optimization, or alternatively using ^ to indicate anchoring explicitly.

When a capturing subpattern is repeated, the value captured is the substring that matched the final iteration. For example, after

```
(tweedle[dume]{3}\s*)+
```

has matched "tweedledum tweedledee" the value of the captured substring is "tweedledee". However, if there are nested capturing subpatterns, the corresponding captured values may have been set in previous iterations. For example, after

```
/(a|(b))+/
```

matches "aba" the value of the second captured substring is

"b".

BACK REFERENCES

Outside a character class, a backslash followed by a digit greater than 0 (and possibly further digits) is a back reference to a capturing subpattern earlier (i.e. to its left) in the pattern, provided there have been that many previous capturing left parentheses.

However, if the decimal number following the backslash is less than 10, it is always taken as a back reference, and causes an error only if there are not that many capturing left parentheses in the entire pattern. In other words, the parentheses that are referenced need not be to the left of the reference for numbers less than 10. See the section entitled "Backslash" above for further details of the handling of digits following a backslash.

A back reference matches whatever actually matched the capturing subpattern in the current subject string, rather than anything matching the subpattern itself. So the pattern

```
(sens|respons)e and \1ibility
```

matches "sense and sensibility" and "response and responsibility", but not "sense and responsibility". If caseful matching is in force at the time of the back reference, then the case of letters is relevant. For example,

```
((?i)rah)\s+\1
```

matches "rah rah" and "RAH RAH", but not "RAH rah", even though the original capturing subpattern is matched caselessly.

There may be more than one back reference to the same subpattern. If a subpattern has not actually been used in a particular match, then any back references to it always fail. For example, the pattern

```
(a|(bc))\2
```

always fails if it starts to match "a" rather than "bc". Because there may be up to 99 back references, all digits following the backslash are taken as part of a potential back reference number. If the pattern continues with a digit character, then some delimiter must be used to terminate the back reference. If the PCRE_EXTENDED option is set, this can

be whitespace. Otherwise an empty comment can be used.

A back reference that occurs inside the parentheses to which it refers fails when the subpattern is first used, so, for example, (a\1) never matches. However, such references can be useful inside repeated subpatterns. For example, the pattern

```
(a|b\1)+
```

matches any number of "a"s and also "aba", "ababaa" etc. At each iteration of the subpattern, the back reference matches the character string corresponding to the previous iteration. In order for this to work, the pattern must be such that the first iteration does not need to match the back reference. This can be done using alternation, as in the example above, or by a quantifier with a minimum of zero.

Assertions

An assertion is a test on the characters following or preceding the current matching point that does not actually consume any characters. The simple assertions coded as \b, \B, \A, \Z, \z, ^ and $ are described above. More complicated assertions are coded as subpatterns. There are two kinds: those that look ahead of the current position in the subject string, and those that look behind it.

An assertion subpattern is matched in the normal way, except that it does not cause the current matching position to be changed. Lookahead assertions start with (?= for positive assertions and (?! for negative assertions. For example,

```
\w+(?=;)
```

matches a word followed by a semicolon, but does not include the semicolon in the match, and

```
foo(?!bar)
```

matches any occurrence of "foo" that is not followed by "bar". Note that the apparently similar pattern

```
(?!foo)bar
```

does not find an occurrence of "bar" that is preceded by something other than "foo"; it finds any occurrence of "bar" whatsoever, because the assertion (?!foo) is always `TRUE` when the next three characters are "bar". A lookbehind

assertion is needed to achieve this effect.

Lookbehind assertions start with (?<= for positive assertions and (?<! for negative assertions. For example,

```
(?<!foo)bar
```

does find an occurrence of "bar" that is not preceded by "foo". The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=bullock|donkey)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error at compile time. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is an extension compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. Lookbehinds in conjunction with once-only subpatterns can be particularly useful for matching at the ends of strings; an example is given at the end of the section on once-only subpatterns.

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?<!999)foo
```

matches "foo" preceded by three digits that are not "999". Notice that each of the assertions is applied independently at the same point in the subject string. First there is a check that the previous three characters are all digits, then there is a check that the same three characters are not

"999". This pattern does not match "foo" preceded by six characters, the first of which are digits and the last three of which are not "999". For example, it doesn't match "123abcfoo". A pattern to do that is

```
(?<=\d{3}...)(?<!999)foo
```

This time the first assertion looks at the preceding six characters, checking that the first three are digits, and then the second assertion checks that the preceding three characters are not "999".

Assertions can be nested in any combination. For example,

```
(?<=(?<!foo)bar)baz
```

matches an occurrence of "baz" that is preceded by "bar" which in turn is not preceded by "foo", while

```
(?<=\d{3}(?!999)...)foo
```

is another pattern which matches "foo" preceded by three digits and any three characters that are not "999".

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If any kind of assertion contains capturing subpatterns within it, these are counted for the purposes of numbering the capturing subpatterns in the whole pattern. However, substring capturing is carried out only for positive assertions, because it does not make sense for negative assertions.

Assertions count towards the maximum of 200 parenthesized subpatterns.

Once-only subpatterns

With both maximizing and minimizing repetition, failure of what follows normally causes the repeated item to be re-evaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern \d+foo when applied to the subject line

```
123456bar
```

After matching all 6 digits and then failing to match "foo", the normal action of the matcher is to try again with only 5 digits matching the \d+ item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be re-evaluated in this way, so the matcher would give up immediately on failing to match "foo" the first time. The notation is another kind of special parenthesis, starting with (?> as in this example:

```
(?>\d+)bar
```

This kind of parenthesis "locks up" the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Backtracking past it to previous items, however, works as normal.

An alternative description is that a subpattern of this type matches the string of characters that an identical standalone pattern would match, if anchored at the current point in the subject string.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both \d+ and \d+? are prepared to adjust the number of digits they match in order to make the rest of the pattern match, (?>\d+) can only match an entire sequence of digits.

This construction can of course contain arbitrarily complicated subpatterns, and it can be nested.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of the subject string. Consider a simple pattern such as

```
abcd$
```

when applied to a long string which does not match. Because matching proceeds from left to right, PCRE will look for each "a" in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as

```
^.*abcd$
```

then the initial .* matches the entire string at first, but when this fails (because there is no following "a"), it backtracks to match all but the last character, then all but

the last two characters, and so on. Once again the search for "a" covers the entire string, from right to left, so we are no better off. However, if the pattern is written as

```
^(?>.*)(?<=abcd)
```

then there can be no backtracking for the .* item; it can match only the entire string. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the match fails immediately. For long strings, this approach makes a significant difference to the processing time.

When a pattern contains an unlimited repeat inside a subpattern that can itself be repeated an unlimited number of times, the use of a once-only subpattern is the only way to avoid some failing matches taking a very long time indeed. The pattern

```
(\D+|<\d+>)*[!?]
```

matches an unlimited number of substrings that either consist of non-digits, or digits enclosed in <>, followed by either ! or ?. When it matches, it runs quickly. However, if it is applied to

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

it takes a long time before reporting failure. This is because the string can be divided between the two repeats in a large number of ways, and all have to be tried. (The example used [!?] rather than a single character at the end, because both PCRE and Perl have an optimization that allows for fast failure when a single character is used. They remember the last single character that is required for a match, and fail early if it is not present in the string.)
If the pattern is changed to

```
((?>\D+)|<\d+>)*[!?]
```

sequences of non-digits cannot be broken, and failure happens quickly.

Conditional subpatterns

It is possible to cause the matching process to obey a subpattern conditionally or to choose between two alternative subpatterns, depending on the result of an assertion, or whether a previous capturing subpattern matched or not. The

two possible forms of conditional subpattern are

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

If the condition is satisfied, the yes-pattern is used; otherwise the no-pattern (if present) is used. If there are more than two alternatives in the subpattern, a compile-time error occurs.

There are two kinds of condition. If the text between the parentheses consists of a sequence of digits, then the condition is satisfied if the capturing subpattern of that number has previously matched. Consider the following pattern, which contains non-significant white space to make it more readable (assume the PCRE_EXTENDED option) and to divide it into three parts for ease of discussion:

```
( \( )?    [^()]+    (?(1) \) )
```

The first part matches an optional opening parenthesis, and if that character is present, sets it as the first captured substring. The second part matches one or more characters that are not parentheses. The third part is a conditional subpattern that tests whether the first set of parentheses matched or not. If they did, that is, if subject started with an opening parenthesis, the condition is `TRUE`, and so the yes-pattern is executed and a closing parenthesis is required. Otherwise, since no-pattern is not present, the subpattern matches nothing. In other words, this pattern matches a sequence of non-parentheses, optionally enclosed in parentheses.

If the condition is not a sequence of digits, it must be an assertion. This may be a positive or negative lookahead or lookbehind assertion. Consider this pattern, again containing non-significant white space, and with the two alternatives on the second line:

```
(?(?=[^a-z]*[a-z])
\d{2}-[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )
```

The condition is a positive lookahead assertion that matches an optional sequence of non-letters followed by a letter. In other words, it tests for the presence of at least one letter in the subject. If a letter is found, the subject is matched against the first alternative; otherwise it is matched against the second. This pattern matches strings in one of the two forms dd-aaa-dd or dd-dd-dd, where aaa are letters and dd are digits.

Comments

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

If the PCRE_EXTENDED option is set, an unescaped # character outside a character class introduces a comment that continues up to the next newline character in the pattern.

Recursive patterns

Consider the problem of matching a string in parentheses, allowing for unlimited nested parentheses. Without the use of recursion, the best that can be done is to use a pattern that matches up to some fixed depth of nesting. It is not possible to handle an arbitrary nesting depth. Perl 5.6 has provided an experimental facility that allows regular expressions to recurse (amongst other things). The special item (?R) is provided for the specific case of recursion. This PCRE pattern solves the parentheses problem (assume the PCRE_EXTENDED option is set so that white space is ignored):

```
\( ( (?>[^()]+) | (?R) )* \)
```

First it matches an opening parenthesis. Then it matches any number of substrings which can either be a sequence of non-parentheses, or a recursive match of the pattern itself (i.e. a correctly parenthesized substring). Finally there is a closing parenthesis.

This particular example pattern contains nested unlimited repeats, and so the use of a once-only subpattern for matching strings of non-parentheses is important when applying the pattern to strings that do not match. For example, when it is applied to

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

it yields "no match" quickly. However, if a once-only subpattern is not used, the match runs for a very long time indeed because there are so many different ways the + and * repeats can carve up the subject, and all have to be tested before failure can be reported.

The values set for any capturing subpatterns are those from the outermost level of the recursion at which the subpattern value is set. If the pattern above is matched against

```
(ab(cd)ef)
```

the value for the capturing parentheses is "ef", which is the last value taken on at the top level. If additional parentheses are added, giving

```
\( ( ( (?>[^()]+) | (?R) )* ) \)
  ^                   ^
  ^                   ^ then the string they capture
```

is "ab(cd)ef", the contents of the top level parentheses. If there are more than 15 capturing parentheses in a pattern, PCRE has to obtain extra memory to store data during a recursion, which it does by using pcre_malloc, freeing it via pcre_free afterwards. If no memory can be obtained, it saves data for the first 15 capturing parentheses only, as there is no way to give an out-of-memory error from within a recursion.

Performances

Certain items that may appear in patterns are more efficient than others. It is more efficient to use a character class like [aeiou] than a set of alternatives such as (a|e|i|o|u). In general, the simplest construction that provides the required behaviour is usually the most efficient. Jeffrey Friedl's book contains a lot of discussion about optimizing regular expressions for efficient performance.

When a pattern begins with .* and the PCRE_DOTALL option is set, the pattern is implicitly anchored by PCRE, since it can match only at the start of a subject string. However, if PCRE_DOTALL is not set, PCRE cannot make this optimization, because the . metacharacter does not then match a newline, and if the subject string contains newlines, the pattern may match from the character immediately following one of them instead of from the very start. For example, the pattern

```
(.*) second
```

matches the subject "first\nand second" (where \n stands for a newline character) with the first captured substring being "and". In order to do this, PCRE has to retry the match starting after every newline in the subject.

If you are using such a pattern with subject strings that do not contain newlines, the best performance is obtained by setting PCRE_DOTALL , or starting the pattern with ^.* to indicate explicit anchoring. That saves PCRE from having to

scan along the subject looking for a newline to restart at.

Beware of patterns that contain nested indefinite repeats. These can take a long time to run when applied to a string that does not match. Consider the pattern fragment

```
(a+)*
```

This can match "aaaa" in 33 different ways, and this number increases very rapidly as the string gets longer. (The * repeat can match 0, 1, 2, 3, or 4 times, and for each of those cases other than 0, the + repeats can match different numbers of times.) When the remainder of the pattern is such that the entire match is going to fail, PCRE has in principle to try every possible variation, and this can take an extremely long time.

An optimization catches some of the more simple cases such as

```
(a+)*b
```

where a literal character follows. Before embarking on the standard matching procedure, PCRE checks that there is a "b" later in the subject string, and if there is not, it fails the match immediately. However, when there is no following literal this optimization cannot be used. You can see the difference by comparing the behaviour of

```
(a+)*\d
```

with the pattern above. The former gives a failure almost instantly when applied to a whole line of "a" characters, whereas the latter takes an appreciable time with strings longer than about 20 characters.

LXXXVI. qtdom functions

qdom_tree (PHP 4 >= 4.0.4)

creates a tree of an xml string

object **qdom_tree** (string)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

qdom_error (PHP 4 >= 4.0.5)

Returns the error string from the last QDOM operation or FALSE if no errors occured

string **qdom_error** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

LXXXVII. Reguläre Ausdrücke Funktionen (POSIX erweitert)

Anmerkung: Bei Verwendung der PCRE Funktionen unterstützt PHP auch Reguläre Ausdrücke mit einer zu Perl kompatiblen Syntax. Diese Funktionen unterstützen non-greedy Matching, Assertions, bedingte Subpatterns und viele andere Merkmale, die von der POSIX-erweiterten Syntax regulärer Ausdrücke nicht unterstützt werden.

Warnung

Diese Funktionen regulärer Ausdrücke sind im Gegensatz zu den PCRE Funktionen nicht binary-safe.

Reguläre Ausdrücke werden für komplexe Manipulationen an Zeichenketten mit PHP verwendet. Folgende Funktionen unterstützen reguläre Ausdrücke:

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()
- spliti()

Alle diese Funktionen nehmen als erstes Argument einen regulären Ausdruck an. PHP verwendet durch POSIX 1003.2 definierte POSIX erweiterte reguläre Ausdrücke. Eine vollständige Beschreibung der POSIX regulären Ausdrücke finden Sie im Regex-Verzeichnis der PHP-Distribution in den Regex man pages. Da sie im manpage-Fomat vorliegt, sollten Sie einen Befehl der Art **man /usr/local/src/regex/regex.7** verwenden, um sie zu lesen.

Beispiel 1. Beispiele regulärer Ausdrücke

```
ereg ("abc", $string);
/* Gibt true zurück, falls "abc"
   irgendwo in $string gefunden wird. */

ereg ("^abc", $string);
/* Gibt true zurück, falls "abc"
   am Anfang von $string gefunden wird. */

ereg ("abc$", $string);
/* Gibt true zurück, falls "abc"
   am Ende von $string gefunden wird. */

eregi ("(ozilla.[23]|MSIE.3)", $HTTP_USER_AGENT);
/* Gibt true zurück, falls es sich beim Client Browser
   um Netscape 2, 3 oder MSIE 3 handelt. */
```

```
ereg ("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $string,$regs);
/* Setzt drei Wörter, die durch Leerzeichen getrennt
   sind, in $regs[1], $regs[2] und $regs[3] ein. */

$string = ereg_replace ("^", "<br />", $string);
/* Setzt ein <br /> Tag vor $string. */

$string = ereg_replace ("$", "<br />", $string);
/* Setzt ein <br /> Tag hinter $string. */

$string = ereg_replace ("\n", "", $string);
/* Entfernt alle Zeilenumbrüche aus $string. */
```

ereg (PHP 3, PHP 4 >= 4.0.0)

Sucht Übereinstimmungen mit einem regulären Ausdruck

int **ereg** (string Suchmuster, string Zeichenkette [, array regs])

> **Anmerkung:** Die Funktion preg_match(), die eine zu Perl kompatible Syntax regulärer Ausdrücke verwendet, ist häufig die schnellere Alternative zu **ereg()**.

Sucht in *`Zeichenkette`* nach Übereinstimmungen mit dem regulären Ausdruck, der in *`Suchmuster`* angegeben wurde.

Wenn Übereinstimmungen mit eingeklammerten Teilzeichenketten von *`Suchmuster`* gefunden werden und die Funktion mit dem dritten Argument *`regs`* aufgerufen wurde, werden die Übereinstimmungen in den Elementen des Arrays *`regs`* gespeichert. $regs[1] enthält dann die Teilzeichenkette der ersten Klammer, $regs[2] die Teilzeichenkette der zweiten usw. $regs[0] enthält bei Übereinstimmung mit *`Zeichenkette`* eine Kopie der kompletten *`Zeichenkette`*.

Bis (und inklusive) PHP 4.1.0 wird `$regs` mit genau zehn Elementen gefüllt, auch wenn es tatsächlich mehr oder weniger Übereinstimmungen mit den eingeklammerten Teilzeichenketten gibt. Dies wirkt sich aber nicht auf **ereg()**s Fähigkeit aus, mehr übereinstimmende Teilzeichenketten zu finden. Falls keine Übereinstimmungen gefunden werden, wird `$regs` nicht von **ereg()** verändert.

Die Suche unterscheidet zwischen Groß und Kleinschreibung.

Gibt `TRUE` zurück, wenn in *`Zeichenkette`* eine Übereinstimmung mit *`Suchmuster`* gefunden wurde oder `FALSE`, wenn keine Übereinstimmung gefunden wurde oder wenn ein Fehler aufgetreten ist.

Der folgende Code-Schnipsel mimmt ein Datum im ISO-Format (JJJJ-MM-TT) und gibt es im Format TT.MM.JJJJ aus:

Beispiel 1. ereg() Beispiel

```
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Ungültiges Datumsformat: $date";
}
```

Siehe auch: eregi(), ereg_replace(), eregi_replace() und preg_match().

ereg_replace (PHP 3, PHP 4 >= 4.0.0)

Ersetzt einen regulären Ausdruck

string **ereg_replace** (string Suchmuster, string Ersatz, string Zeichenkette)

> **Anmerkung:** Die Funktion preg_replace(), die eine zu Perl kompatible Syntax regulärer Ausdrücke verwendet, ist häufig die schnellere Alternative zu **ereg_replace()**.

Diese Funktion durchsucht *`Zeichenkette`* nach Übereinstimmungen mit *`Suchmuster`* und ersetzt dann den übereinstimmenden Text durch *`Ersatz`*.

Zurückgegeben wird die geänderte Zeichenkette, was auch bedeuten könnte, dass die ursprüngliche Zeichenkette zurückgegeben wird, wenn es keine zu ersetzenden Übereinstimmungen gibt.

Wenn *`Suchmuster`* eingeklammerte Teilzeichenketten enthält, darf *`Ersatz`* Teilzeichenketten der Form `\\`*`Ziffer`* enthalten, die dann durch die Ziffer'ste Teilzeichenkette ersetzt werden. `\\0` erzeugt den gesamten Inhalt der durchsuchten Zeichenkette. Bis zu neun Teilzeichenketten dürfen verwendet werden. Klammern dürfen geschachtelt werden und werden in diesem Fall anhand der öffnenden Klammern gezählt.

Wenn in *`Zeichenkette`* keine Übereinstimmungen gefunden werden, wird *`Zeichenkette`* unverändert zurückgegeben.

Der folgende Code-Schnipsel gibt zum Beispiel dreimal "Das war ein Test" aus:

Beispiel 1. ereg_replace() Beispiel

```
$string = "Das ist ein Test";
echo ereg_replace (" ist", " war", $string);
echo ereg_replace ("( )ist", "\\1war", $string);
echo ereg_replace ("(( )ist)", "\\2war", $string);
```

Folgendes ist zu beachten: Wenn Sie als *`Ersatz`* einen ganzzahligen Wert verwenden, bekommen sie vielleicht nicht das erwartete Ergebnis. Das liegt daran, dass **ereg_replace()** die Zahl als Ordinalwert eines Zeichens betrachtet und dieses verwendet. Zum Beispiel:

Beispiel 2. ereg_replace() Beispiel

```
<?php
/* Dieses funktioniert nicht wie erwartet. */
$zahl = 4;
$zeichenkette = "Diese Zeichenkette hat vier Wörter.";
$zeichenkette = ereg_replace('vier', $zahl, $zeichenkette);
echo $zeichenkette;   /* Ausgabe: 'Diese Zeichenkette hat   Wörter.' */

/* Dieses funktioniert. */
$zahl = '4';
$zeichenkette = "Diese Zeichenkette hat vier Wörter.";
$zeichenkette = ereg_replace('vier', $zahl, $zeichenkette);
echo $zeichenkette;   /* Output: 'Diese Zeichenkette hat 4 Wörter.' */
?>
```

Beispiel 3. Ersetzen von URLs durch Links

```
$text = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
                     "<a href=\"\\0\">\\0</a>", $text);
```

Siehe auch: ereg(), eregi(), eregi_replace(), str_replace() und preg_match().

eregi (PHP 3, PHP 4 >= 4.0.0)

Sucht Übereinstimmung mit regulärem Ausdruck ohne Berücksichtigung von Groß-/Kleinschreibung

int **eregi** (string Suchmuster, string Zeichenkette [, array regs])

Diese Funktion entspricht ereg() mit dem Unterschied, dass sie übereinstimmende Buchstaben nicht nach Groß- und Kleinschreibung unterscheidet.

Beispiel 1. eregi() Beispiel

```
if (eregi("z", $string)) {
    echo "'$string' contains a 'z' or 'Z'!";
}
```

Siehe auch: ereg(), ereg_replace() und eregi_replace().

eregi_replace (PHP 3, PHP 4 >= 4.0.0)

Ersetzt einen regulären Ausdrück ohne Berücksichtigung von Groß-/Kleinschreibung

string **eregi_replace** (string Suchmuster, string Ersatz, string Zeichenkette)

Diese Funktion entspricht ereg_replace() mit dem Unterschied, dass sie übereinstimmende Buchstaben nicht nach Groß- und Kleinschreibung unterscheidet.

Siehe auch: ereg(), eregi() und ereg_replace().

split (PHP 3, PHP 4 >= 4.0.0)

Zerlegt eine Zeichenkette anhand eines regulären Ausdrucks in ein Array

array **split** (string Suchmuster, string Zeichenkette [, int Beschränkung])

> **Anmerkung:** Die Funktion preg_split(), die eine zu Perl kompatible Syntax regulärer Ausdrücke verwendet, ist häufig die schnellere Alternative zu **split()**.

Gibt ein Array mit Zeichenketten zurück, die jeweils eine Teilzeichenkette von *`Zeichenkette`* sind und durch deren Zerlegung an den durch *`Suchmuster`* des regulären Ausdrucks bestimmten Stellen entstehen. Wenn *`Beschränkung`* gesetzt wurde, enthält das zurückgegebene Array höchstens *`Beschränkung`* Elemente, von denen das letzte den ganzen Rest von *`Zeichenkette`* enthält. Bei Auftreten eines Fehlers gibt **split()** `FALSE` zurück.

Um die ersten vier Felder einer Zeile von `/etc/passwd` abzuspalten:

Beispiel 1. split() Beispiel

```
list($user,$pass,$uid,$gid,$extra)= split (":", $passwd_line, 5);
```

> **Anmerkung:** Wenn *`Suchmuster`* *`n`* mal vorkommt, enhält das zurückgegebene Array *`n`*`+1` Elemente. Kommt *`Suchmuster`* zum Beispiel überhaupt nicht vor, wird ein Array mit nur einem Element zurückgegeben. Das gilt natürlich auch, wenn *`Zeichenkette`* leer ist.

Um ein Datum zu analysieren, das durch Schrägstriche, Punkte oder Bindestriche getrennt ist:

Beispiel 2. split() Beispiel

```
$datum = "04/30/1973";  // Trennzeichen darf ein Schrägstrich, Punkt
                        // oder Bindestrich sein
list ($monat, $tag, $jahr) = split ('[/.-]', $datum);
echo "Monat: $monat; Tag: $tag; Jahr: $jahr<br>\n";
```

Beachten Sie, dass *`Suchmuster`* Groß- und Kleinschreibung berücksichtigt.

Beachten Sie: wenn Sie die Fähigkeiten regulärer Ausdrücke nicht benötigen, ist die Verwendung von explode() schneller, weil diese Funktion nicht unter der Last der Engine für reguläre Ausdrücke steht.

Wenn Sie nach einer Möglichkeit suchen, das Verhalten von Perls **$chars = split(", $str)** nachzuahmen, schauen Sie sich bitte die Beispiele für preg_split() an.

Bitte beachten Sie, dass *`Suchmuster`* ein regulärer Ausdruck ist. Wenn Sie eine Zerlegung an Zeichen vornehmen, die in regulären Ausdrücken als besondere Zeichen betrachtet werden, müssen Sie diese entsprechend kennzeichnen. Wenn Sie der Ansicht sind, **split()** (oder, bei diesem Thema, eine andere Regex-Funktion) verhalte sich seltsam, lesen Sie bitte die Datei `regex.7`, die sich im `regex/` Unterverzeichnis ihrer PHP-Distribution befindet. Da sie im manpage-Fomat vorliegt, sollten Sie einen Befehl der Art **man /usr/local/src/regex/regex.7** verwenden, um sie zu lesen.

Siehe auch: preg_split(), spliti(), explode(), implode(), chunk_split() und wordwrap().

spliti (PHP 4)

Zerlegt eine Zeichenkette anhand eines regulären Ausdrucks ohne Berücksichtigung von Groß-/Kleinschreibung in ein Array

array **spliti** (string Suchmuster, string Zeichenkette [, int Beschränkung])

Diese Funktion entspricht split() mit dem Unterschied, dass sie übereinstimmende Buchstaben nicht nach Groß- und Kleinschreibung unterscheidet.

Siehe auch: **preg_spliti()**, split(), explode() und implode().

sql_regcase (PHP 3, PHP 4 >= 4.0.0)

Erstellt einen regulären Ausdruck für eine Suche nach Übereinstimmungen ohne Berücksichtigung von Groß-/Kleinschreibung

string **sql_regcase** (string Zeichenkette)

Gibt einen korrekten regulären Ausdruck zurück, der mit *`Zeichenkette`* ohne Berücksichtigung von Groß-/Kleinschreibung übereinstimmt. Dieser Ausdruck entspricht *`Zeichenkette`*, wobei jedes Zeichen in einen Klammerausdruck umgewandelt wird. Die Klammer enthält dieses Zeichen, wenn möglich, in groß- und kleingeschriebener Form, andernfalls enthält sie das ursprüngliche Zeichen zweimal.

Beispiel 1. sql_regcase() Beispiel

```
echo sql_regcase ("Foo bar");
```

Gibt

```
[Ff][Oo][Oo] [Bb][Aa][Rr]
```

aus.

Dies kann bei Produkten, die nur reguläre Ausdrücke mit Berücksichtigung von Groß-/Kleinschreibung unterstützen, verwendet werden, um eine Übereinstimmung mit Suchmustern ohne Berücksichtigung von Groß-/Kleinschreibung zu erreichen.

LXXXVIII. Semaphor und Shared Memory Funktionen

Dieses Modul bietet Unterstützung für die Nutzung von Semaphoren und gemeinamen Speichersegmenten mit Hilfe der UNIX System V Mechanismen zur Interprozess-Kommunikation.

Mit Hilfe von Semaphoren kann der exklusive Zugriff auf Resourcen eines Systems sichergestellt oder die Anzahl der Prozesse begrenzt werden, die gleichzeitig eine bestimmte Systemresource nutzen.

Mit Hilfe gemeinsamer Speichersegmente (shared memory) können system-globale Variablen angelegt und bereitgestellt werden. Diese Speicherbereiche stehen allen Prozessen des Systems, insbesondere aber allen parallel laufenden PHP-Instanzen zum Datenaustausch zur Verfügung. Beachten Sie bitte, das gemeinsame Speichersegmente keinen Schutze vor konkurierenden gleichzeitigen Zugriffen bieten. Benutzen Sie Semaphore für die Koordination solcher Zugriffe.

Tabelle 1. Limits of Shared Memory by the Unix OS

SHMMAX	Maximalgröße eines gemeinsamen Speicherbereichs, üblich sind 131072 Byte (128KB)
SHMMIN	Minimalgröße eines gemeinsamen Speicherbereichs, normalerweise 1 Byte
SHMMNI	maximale Anzahl unterschiedlicher Speichersegmente, üblich sind 100
SHMSEG	Anzahl der Speichersegmente, die ein einzelner Prozess einbinden darf, üblich sind 6

Beispiel 1. Koordinierter Zugriff auf gemeinsamen Speicher

```
<?php
// gemeinsames Speichersegment beschaffen
if(! ($mkey = shm_attach(0x2328,1024,OctDec("666"))))
{ echo "shmem_attach fehlgeschlagen<br>\n"; exit;}

// Semaphor für Zugriffskoordination auf
// Speichersegment beschaffen
if(! ($skey = sem_get(0x2328,1,OctDec("666"))))
{ echo "sem_get fehlgeschlagen<br>\n"; exit;}

// Zugriff anfordern
if(! sem_acquire($skey))
{ echo "sem_acquire fehlgeschlagen<br>\n"; exit;}

// Wert lesen, aktualisieren und schreiben
$val = @shm_get_var($mkey,1);
if($val===false) $val=1; else $val++;
print "new value is $val<br>\n";
shm_put_var($mkey,1,$val);
```

```
// Zugriff freigeben
sem_release($skey);
?>
```

sem_get (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liefert ein Semaphore-Handle

int **sem_get** (int key [, int max_acquire [, int perm]])

Diese Funktion liefert ein Handle, das für den Zugriff auf das Semaphore mit der Kennung *key* benutzt werden kann.

Falls noch kein Semaphore mit dieser Kennung existiert, so wird es neu angelegt. Die maximale Anzahl der gleichzeitig möglichen Anforderungen und die Zugriffsrechte werden dabei über die Parameter *max_aquire* und **sem_get()** festgelegt.

Wird der angeforderte Semaphore zum Zeitpunkt des Aufrufs von keiner anderen Funktion genutzt, so kann über den optionalen Parameter *max_aquire* die Anzahl der gelichzeitig möglichen Anforderungen neu festgelegt werden, anderenfalls bleibt der ursprüngliche Wert erhalten.

Ein zweiter Aufruf der Funktion mit demselben *key* liefert ein weiteres, von dem ersten verschiedenes, Handle, das aber trotzdem denselben darunterliegenden Semaphore anspricht.

Wenn kein Semaphore mit der Kennung *key* angelegt werden kann oder die Rechte für den Zugriff auf einen bestehendennicht ausreichen, so ist das Ergebnis dieser Funktion der Wert `FALSE`.

Siehe auch sem_acquire() und sem_release().

sem_acquire (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zugriff auf einen Semaphore anfordern

int **sem_acquire** (int sem_identifier)

sem_acquire() blockiert die Ausführung (falls notwendig) solange, bis eine Anforderung eines Semaphores durchgeführt werden kann, d.h. solange die Anzahl der gleichzeitigen Anforderungen bereits der maximalen Anzahl entspricht.

Wird ein erfolgreich angeforderter Sempahore innerhalb eines Requests nicht wieder freigegeben, so erfolgt die Freigabe nach Abarbeitung des Requests automatisch und es wird eine Warnung ausgegeben.

Diese Funktion liefert `TRUE` bei Erfolg und `FALSE`, wenn eine Anforderung des Semaphores wegen eines Fehlers nicht möglich ist.

Siehe auch sem_get() und sem_release().

Warnung

Wird ein Semaphore ein zweites mal während desselben Request angefordert, ohne zwischenzeitlich freigegeben worden zu sein, so kann dies dazu führen, das die weitere Ausführung dieses Requests nicht mehr möglich ist.

sem_release (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Freigabe eines angeforderten Semaphores

int **sem_release** (int sem_identifier)

sem_release() gibt einen zuvor angeforderten Semaphore wieder frei. Die Anforderung und Freigabe kann innerhalb eines Prozesses mehrfach hintereinander erfolgen.

Die Funktion liefert `TRUE`, wenn ein angeforderter Semaphore erfolgreich freigegeben wurde, ansonsten `FALSE`.

Siehe auch sem_get() und sem_acquire().

shm_attach (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Anlegen oder anbinden eines gemeinsamen Speichersegments

int **shm_attach** (int key [, int memsize [, int perm]])

shm_attach() gibt ein Handle für den Zugriff auf den gemeinsamen Speicherbereich mit der Kennung *`key`* zurück.

Falls noch kein Bereich mit dieser Kennung existiert, so wird er mit einer Größe von *`memsize`* Byte (Vorgabe ist der Parameter *`sysvshm.init_mem`* aus der Konfigurationsdatei bzw. 10000 Byte) und den Zugriffsberechtigungen gemäß *`perm`* (Vorgabe ist der Wert 0666, der Lese- und Schreibzugriffe für alle Benutzer gestattet) an.

Ein zweiter Aufruf der Funktion mit demselben *`key`* liefert ein weiteres, von dem ersten verschiedenes, Handle, das aber trotzdem denselben gemeinsamen Speicherbereich anspricht.

Wenn kein Bereich mit der Kennung *`key`* angelegt werden kann oder die Rechte für den Zugriff auf ein bestehendes Segment nicht ausreichen, so ist das Ergebnis dieser Funktion der Wert `FALSE`.

shm_detach (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Beenden der Anbindung an ein gemeinsames Speichersegment

int **shm_detach** (int shm_identifier)

shm_detach() beendet die Anbindung eines gemeinsamen Speicherbreichs mit dem Handle *`shm_identifier`*, das zuvor über shm_attach() angebunden wurde. Das Segment an sich und die darin enthaltenen Daten bleiben erhalten und stehen nach erneuter Anbindung in diesem oder einem anderen Prozess wieder zur Verfügung.

shm_remove (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Entfernt ein gemeinsames Speichersegment

int **shm_remove** (int key)

Diese Funktion gibt ein gemeinsames Speichersegment wieder frei, dabei gehen alle Daten innerhalb des Segmentes verloren.

Diese Funktion liefert `TRUE`, wenn das Segment erfolgreich entfernt wurde, oder `FALSE`, wenn kein Segment mit der Kennung *`key`* existierte oder die Zugriffsrechte auf ein bestehendes Segment nicht für ein Löschen ausreichen.

shm_put_var (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Aktualisiert einen in einem gemeinsamen Speicherbereich angelegten Wert.

int **shm_put_var** (int shm_identifier, int variable_key, mixed variable)

Diese Funktion weist einer Variable mit der Kennung *`variable_key`* in einem gemeinsamen Speichersegment den Wert von *`variable`* zu, beim ersten Zugriff wird *`variable_key`* neu angelegt.

Diese Funktion liefert `FALSE` zurück, wenn der verbleibende Speicherplatz im angegebenen Segment *`shm_identifier`* zur Aufnahme des Wertes von *`variable`* nicht mehr ausreicht, ansonsten `TRUE`.

shm_get_var (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Liest einen in einem gemeinsamen Speicherbereich angelegten Wert.

mixed **shm_get_var** (int id, int variable_key)

shm_get_var() liefert den Wert mit der Kennung *`variable_key`* aus einem gemeinsamen Speicherbereich. Der Wert im gemeinsamen Speicher bleibt weiter vorhanden. Existiert kein Wert mit der angegebenen Kennung, so wird eine Warnung erzeugt und `FALSE` zurückgegeben.

shm_remove_var (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Entfernt einen in einem gemeinsamen Speicherbereich angelegten Wert.

int **shm_remove_var** (int id, int variable_key)

Diese Funktion entfernt den Wert mit der Kennung *`variable_key`* aus einem gemeinsamen Speichersegment und stellt den hiervon genutzten Speicher innerhalb des egmentes wieder zur Verfügung.

Diese Funktion liefert `TRUE` bei Erfolg und `FALSE`, falls kein Wert mit der Kennung *`variable_key`* im Speichersegment angelegt war.

LXXXIX. SESAM Datenbankfunktionen

SESAM/SQL-Server ist ein von Fujitsu Siemens Computers in Deutschland entwickeltes Mainframe-Datenbanksystem für High-End Server mit dem Betriebssystem BS2000/OSD.

In zahlreichen BS2000-Installationen hat SESAM/SQL-Server unter Beweis gestellt:

- die einfach zu handhabende Unterstützung von Java-, Web- und Client-Server-Anwendungen
- eine Hochverfügbarkeit von mehr als 99,99 %
- die Fähigkeit, bis zu mehrere hunderttausend Benutzer zu bedienen

Mit einer PHP3 SESAM-Schnittstelle sind jetzt auch Datenbank-Operationen per PHP-scripts möglich.

Konfigurationshinweise: Die PHP-SESAM-Schnittstelle muß als Modul in Apache integriert werden. Stand-alone-Betrieb ist nicht möglich. Im Apache-PHP-Modul ist die SESAM-Schnittstelle entsprechend den Apache-Vorschriften zu konfigurieren.

Tabelle 1. SESAM-Konfigurationsvorschriften

Directive	Bedeutung
`php3_sesam_oml`	Name der BS2000-PLAM-Bibliothek mit den ladbaren Modulen des SESAM-Treibers. Für die Nutzung von SESAM-Funktionen erforderlich. Beispiel: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`
`php3_sesam_configfile`	Name der Konfigurationsdatei der SESAM-Anwendung. Für die Nutzung von SESAM-Funktionen erforderlich. Example: `php3_sesam_configfile $SESAM.SESAM.CONF.AW` enthält im allgemeinen folgende Konfiguration (siehe auch SESAM Referenzhandbuch): `CNF=B` `NAM=K` `NOTYPE`
`php3_sesam_messagecatalog`	Name des SESAM-Meldungskatalogs. Nur erforderlich, wenn der SESAM-Meldungskatalog nicht im BS2000-Meldungkatalog enthalten ist. Example: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

Zusätzlich zur PHP-SESAM-Schnittstelle muß natürlich die SESAM-Datenbank selbst auf dem

BS2000-System installiert und konfiguriert werden:

- der SESAM database handler (DBH) muß gestartet sein
- die Datenbank und der DBH müssen verbunden sein

Für eine Verbindung zwischen dem PHP script und dem DBH müssen die Parameter von `CNF` und `NAM` der SESAM-Konfiguration den Werten des im BS2000 gestarteten DBH ensprechen.

Bei verteilten Datenbanken muß ein SESAM/SQL-DCN-Agent gestartet werden, dessen Konfigurationstabelle Host- und Datenbank-Namen enthält.

Die Kommunication zwischen PHP im BS2000-Subsystem POSIX und dem DBH im BS2000 erfolgt über Zugriffe eines speziellen Treibermoduls (SQLSCI) und der SESAM-Anschlußmodule auf einen gemeinsamen Speicherbereich (common memory). Der common-memory-pool-Mechanismus und die Tatsache, daß PHP in den Webserver statisch eingebunden ist, machen die Datenbankzugriffe sehr schnell. Datenbankzugriffe über ODBC, JDBC or UTM sind nicht erforderlich.

Nur ein kleines Lademodul (SESMOD) wird zum PHP gebunden, während die SESAM-Anschlußmodule aus der SESAM-OML (PLAM-Bibliothek) nachgeladen werden. library. In der PHP-Konfiguration,muß diese SESAM-OML und der für die SESAM-Konfiguration erforderliche Linkname angegeben werden (SQLSCI ist wie in SESAM V3.0 in der Standard-SESAM-Toolbibliothek verfügbar).

Wegen der SQL-Syntaxvorschriften für Anführungszeichen (doppelte Anführungszeichen statt AnfÜhrungszeichen mit vorangestelltem Gegenschrägstrich wie in manchen anderen Datenbanken) wird empfohlen, die PHP-Konfigurationsvorschriften `php3_magic_quotes_gpc` und `php3_magic_quotes_sybase` für alle SESAM betreffenden PHP-scripts auf `On` zu setzen.

Überlegungen zur Ablaufzeit: Wegen der spezifischen Eigenschaften der BS2000-Prozeßsteuerung kann der Treiber erst nach dem fork des Apache-Servers für seine Sohn-Prozesse geladen werden. Der erste SESAM-Zugriff jedes dieser Prozesse wird dadurch etwas verlangsamt, alle weiteren Zugriffe werden jedoch ohne Beeinträchtigung abgearbeitet.

Ein explizit definierter SESAM-Meldungskatalog wird bei jedem Laden des Treibers (d.h. beim ersten SESAM-Zugriff) geladen. BS2000 erzeugt nach dem erfolgreichen Laden dieses Katalogs eine Meldung, die an die error_logfile von Apache geschickt wird. Da BS2000 z. Zt. ein Unterdrücken dieser Meldung nicht erlaubt, wird das logfile langsam damit vollgeschrieben.

Bitte beachten: die SESAM-OML (PLAM-Bibliothek) und die SESAM-Konfigurationsdatei müssen für die Benutzerkennung, unter der der Webserver läuft, lesbar sein! Andernfalls kann der Server den Treiber nicht laden, womit die SESAM-Funktionen nicht zur Verfügung stehen. Auch der Zugriff die Datenbank selbst muß dem Server möglich sein, da sonst keine Verbindung zum DBH möglich ist.

Cursor Typen: Die für SQL-Abfragen "select type" zugewiesenen Ergebniscursor können "sequential" oder "scrollable" sein, Standardwert ist wegen des geringeren Speicherbedarfs "sequential"

"Scrollable" Cursors können im Ausgabedatensatz beliebig positioniert werden. Für jede "scrollable" Abfrage gibt es globale Standardwerte für den Scroll-Typ (initialisiert zu: `SESAM_SEEK_NEXT`) und der Scrolling-Abstand, der entweder einmal mit: sesam_seek_row() oder jedesmal beim Lesen einer Zeile mit: sesam_fetch_row() gesetzt werden kann. Die

folgende Tabelle zeigt die Behandlung der globalen Stadtardwerte für Scroll-Typ und Scroll-Abstand beim Lesen einer Zeile mit "scrollable" Cursor:

Tabelle 2. Scrolled Cursor Post-Processing

Scroll Type	Action
`SESAM_SEEK_NEXT`	none
`SESAM_SEEK_PRIOR`	none
`SESAM_SEEK_FIRST`	set scroll type to `SESAM_SEEK_NEXT`
`SESAM_SEEK_LAST`	set scroll type to `SESAM_SEEK_PRIOR`
`SESAM_SEEK_ABSOLUTE`	Auto-Increment internal offset value
`SESAM_SEEK_RELATIVE`	none. (maintain global default *`offset`* value, which allows for, e.g., fetching each 10th row backwards)

Bemerkungen zur Portierung: Da in PHP Indizes üblicherweise mit `NULL` gestartet werden, mußte die SESAM-Schnittstelle entsprechend adaptiert werden: während in der "normalen" SESAM-Schnittstelle ein Array mit Index 1 beginnt, wird in PHP `NULL` als Startpunkt verwendet - z.B. hat beim Lesen von Spalten mit sesam_fetch_row() die erste Spalte den Index 0 und the folgenden Spalten haben Indizes bis zu (aber nicht einschließlich!) dem Spaltenindex ($array["count"]). Beim Portieren von SESAM-Anwendungen aus aus anderen höheren Programmiersprachen nach PHP ist diese Schnittstellenänderung zu beachten. Wo es erforderlich ist, beinhaltet die jeweilige PHP-SESAM-Funktion einen Hinweis auf die mit `NULL` beginnende Indizierung.

Sicherheitsaspekte: Beim Zugriff auf SESAM-Datenbanken sollte der normale Benutzer eines Web-Servers nur die allernötigsten Privilegien bekommen. Im allgemeinen sollte auf Datenbanken nur lesender Zugriff möglich sein. Weitergehende Rechte sollten - nur falls wirklich erforderlich - abhängig vom jeweiligen Einsatzfall eingerichtet werden. Auf keinen Fall sollte man beliebigen Benutzern im Internet volle Zugriffsrechte auf eine Datenbank gewähren! Der Zugriff auf PHP-scripts für die Datenbank-Administration sollte in jedem Fall per Passwort und/oder durch SSL-Verbindung geschützt sein.

Migration von anderen SQL-Datenbanken: Unterschiedliche SQL-Ausprägungen sind nie 100-prozentig kompatibel. Bei der Portierung von SQL-Anwendungen anderer Datenbanken nach SESAM können daher Anpassungen erforderlich sein. Auf folgende, typische Unterschiede sei besonders hingewiesen:

- Herstellerspezifische Datentypen

  Manche herstellerspezifischen Datentypen müssen u.U. durch Standard-SQL-Typen ersetzt werden, wie z.B. `TEXT` in `VARCHAR(max. size)`.

- Schlüsselwörter als SQL-Bezeichner

  In SESAM müssen Schlüsselwörter in Anführungsstrichen (???) angegeben werden

(SQL-Standard) Schlüssel.

- Länge der Anzeige in Datentypen

  Datentypen in SESAM sind nicht durch Längenangabe, sondern durch ihre Genauigkeit spezifiziert. Statt `int(4)` für ganze Zahlen bis '9999' braucht SESAM lediglich `int` für eine implizite Länge von 31 Bit. Als Datentypen für Datum und Zeit gibt es in SESAM nur: `DATE`, `TIME(3)` und `TIMESTAMP(3)`.

- SQL-Datentypen mit herstellerspezifischen Attributen der Form: `unsigned`, `zerofill`, oder `auto_increment` attributes

  `Unsigned` und `zerofill`werden nicht unterstützt. Um die SESAM-implizite automatische Erhöhung zu nutzen, gibt es anstelle von `"... VALUES(0, ...)"` automatisch `Auto_increment` (`"INSERT ... VALUES(*, ...)"`..

- **int ... DEFAULT '0000'**

  Numerische Variablen dürfen nicht mit Zeichenkonstanten initialisiert werden. Statt dessen ist **DEFAULT 0** zu verwenden. Dem Initialisierungs-string für Variablen des SQL-Datentyps datetime muß das jeweilige Schlüsselwort als Prefix vorangestellt werden wie in `CREATE TABLE exmpl ( xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL );`

- **$count = xxxx_num_rows();**

  Manche Datenbanken liefern die geschätzte/erratene Anzahl von Ausgabezeilen als Abfrageresultat an, obwohl der gelieferte Wert grob falsch ist. SESAM kennt die Anzahl von Ausgabezeilen erst, wenn sie tatsächlich gelesen wurden. Falls diese Information TATSÄCHLICH benötigt wird, sollte man **SELECT COUNT(...) WHERE ...**, versuchen. Diese Funktion ermittelt die Anzahl der Treffer. Ein zweiter Aufruf liefert (hoffentlich) die eigentlichen Ergebnisse.

- **DROP TABLE thename;**

  In SESAM muß im Kommamdo **DROP TABLE** der Name entweder um die Schlüsselwörter `RESTRICT` oder `CASCADE` ergänzt werden. Bei der Angabe von `RESTRICT` wird ein Fehler gemeldet, wenn abhängige Objekte (z.B. VIEWs) existieren, während bei `CASCADE` abhängige Objekte zusammen (???) mit der definierten Tabelle gelöscht werden.

Bemerkungen zu weiteren SQL-Datentypen: SESAM unterstützt z.Zt. keine Daten von Typ BLOB. Für zukünftige SESAM-Versionen ist diese Unterstützung geplant.

Bei SQL-Abfragen werden von der PHP-Schnittstelle folgende Datentypen automatisch konvertiert:

Tabelle 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	"integer"

SQL Type	PHP Type
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	"double"
DATE, TIME, TIMESTAMP	"string"
VARCHAR, CHARACTER	"string"

Bei Abfragen auf ganze Zeilen wird das Ergebnis als Array geliefert. Leere Felder werden nicht gefüllt, d.h. ihre Existenz muß explizit mit isset() oder empty() überprüft werden. Dies ermöglicht dem Benutzer eine bessere Kontrolle als durch Rückgabe eines Leer-strings zur Darstellung eines leeren Feldes.

Unterstützung der SESAM-Features "multiple fields": Diese spezielle SESAM-Funktion erlaubt die Definiton von Spalten als Feld-Array. Eine "multiple field" Spalte wird wie folgt definiert:

Beispiel 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test
(
     pkey  CHAR(20) PRIMARY KEY,
     multi(3)   CHAR(12)
)
```

und folgendermaßen ausgefüllt:

Beispiel 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test ( pkey, multi(2..3) )
  VALUES ( 'Second', <'first_val','second_val'>)
```

Zur Beachtung: führende leere Unterfelder (wie in diesem Fall) werden ignoriert und die enthaltenen Werte damit verworfen, so daß in obigem Beispiel das Ergebnis als multi(1..2) angezeigt wird und nicht als multi(2..3).

Bei der Ausgabe von Ergebniszeilen wird auf "multiple columns" wie auf "inlined" zusätzliche Spalten zugegriffen. Im obigen Beispiel bekommt "pkey" den Indexwert `NULL` und auf die drei Spalten "multi(1..3)" kann mit den Indizes 1 bis 3 zugegriffen werden.

SESAM-spezifische Details sind in den Handbüchern the SESAM/SQL-Server documentation (english) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) oder the SESAM/SQL-Server documentation (german) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm) beschrieben. Beide Manuale sind online abrufbar und stehen in gedruckter Form zur Verfügung.

sesam_connect (PHP 3 CVS only)

Eröffnet die Verbindung zu einer SESAM-Datenbank

bool **sesam_connect** (string catalog, string schema, string user)

Ergibt `TRUE` beim erfolgreichen Verbindungsaufbau oder `FALSE` im Fehlerfall.

sesam_connect() baut die Verbindung zu einem SESAM-DBH-Prozeß auf. Diese Verbindung ist "permanent", d.h. nur der erste Zugriff veranlaßt das Laden des Treibers aus der konfigurierten SESAM-OML-Plam-Bibliothek. Alle folgenden Zugriffe nutzen den verwenden den geladenen Treiber und damit automatisch den dadurch definierten Katalog, das Datenbankschema und den Benutzer.

Beim Erstellen einer Datenbank wird der *`"catalog"`* definiert in der SESAM-Konfigurationsvorschrift **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

`"schema"` beschreibt das gewünschte Datenbankschema (siehe SESAM-Benutzerhandbuch).

`"user"` bezieht sich auf Benutzer, die auf die *`"catalog"`* / *`"schema"`* Kombination Zugriff haben. Hinweis: *`"user"`* ist sowohl von der Benutzerkennung im System als auch von der HTTP-Benutzerkennung völlig unabhängig und erscheint nur in der SESAM-Konfiguration.

siehe auch sesam_disconnect().

Beispiel 1. Connect to a SESAM database

```
<?php
if (! sesam_connect ("mycatalog", "myschema", "otto")
     die("Unable to connect to SESAM";
?>
```

sesam_disconnect (PHP 3 CVS only)

Trennen der Verbindung zu einer SESAM-Datenbank

bool **sesam_disconnect** (void)

Rückgabewert: immer `TRUE`.

sesam_disconnect() schließt die logische Verbindung zur SESAM-Datenbank (ohne Verbindungsabbau und Entladen des Treibers).

Hinweis: dieser Funktionsaufruf ist normalerweise nicht erforderlich, da eine offene logische Verbindung automatisch am Ende der Bearbeitung des Skripts geschlossen wird. Ungesicherte Daten werden verworfen, da ein implizites sesam_rollback() durchgeführt wird.

sesam_disconnect() schließt nicht die permanente Verbindung, sondern setzt die Gültigkeit der aktuellen Definition von *`"catalog"`*, *`"schema"`* und *`"user"`* zurück. SESAM-Aufrufe nach **sesam_disconnect()** sind nicht mehr möglich.

Siehe auch: sesam_connect().

Beispiel 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
... some queries and stuff ...
sesam_disconnect(); }
```

sesam_settransaction (PHP 3 CVS only)

Parameter der SESAM-Transaction setzen

bool **sesam_settransaction** (int isolation_level, int read_only)

Ergibt `TRUE` für gültige Werte und wenn **settransaction()** erfolgreich war, ansonsten `FALSE`.

sesam_settransaction() überschreibt die Standardwerte für "isolation level" Parameter und die in der SESAM-Konfigurationsdatei gesetzten "read-only" Transaktionsparameter, um nachfolgende Abfragen zu optimieren und die Konsistenz der Datenbank sicherzustellen. Die geänderten (überschriebenen) Werte wirken nur auf die nächste Transaktion.

sesam_settransaction() kann nur vor dem Start aufgerufen werden, also nicht, wenn eine Transaktion bereits gestartet wurde.

Um die Anwendung in PHP-Scripts zu vereinfachen, wurden folgende Konstanten in PHP vordefiniert (zur Erläuterung der Semantik siehe SESAM-Benutzerhandbuch):

Tabelle 1. Gültige Werte für *"Isolation_Level"* Parameter

Value	Constant	Meaning
1	`SESAM_TXISOL_READ_UNCOMMITTED`	Read Uncommitted
2	`SESAM_TXISOL_READ_COMMITTED`	Read Committed
3	`SESAM_TXISOL_REPEATABLE_READ`	Repeatable Read
4	`SESAM_TXISOL_SERIALIZABLE`	Serializable

Tabelle 2. Gültige Werte für *"Read_Only"* Parameter

Value	Constant	Meaning
0	`SESAM_TXREAD_READWRITE`	Read/Write
1	`SESAM_TXREAD_READONLY`	Read-Only

Die von **sesam_settransaction()** gesetzten Werte überschreiben die in der SESAM-Konfigurationsdatei gesetzten Standardwerte.

Beispiel 1. Setzen von SESAM-Transaktionsparametern

```
<?php
sesam_settransaction(SESAM_TXISOL_REPEATABLE_READ,
                     SESAM_TXREAD_READONLY);
?>
```

sesam_commit (PHP 3 CVS only)

Schreibt noch nicht gesicherte Änderungen in die SESAM-Datenbank

bool **sesam_commit** (void)

Rüchgabewert: `TRUE` bei erfolgreicher Ausführung, ansonsten `FALSE`

sesam_commit() schreibt alle noch ausstehenden Änderungen in die Datenbank

Hinweis: es gibt keine Funktion "automatisches Sichern" wie in anderen Datenbanken, da dies zu ungewolltem Datenverlust führen kann. Bei der Beendigung des aktuellen Skripts oder beim Aufruf von sesam_disconnect()) werden nicht gesicherte Daten durch einen impliziten sesam_rollback() Aufruf verworfen.

Siehe auch: sesam_rollback().

Beispiel 1. Schreiben eines Update in die SESAM-Datenbank

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
  if (!sesam_execimm("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
     die("insert failed");
  if (!sesam_commit())
     die("commit failed");
}
?>
```

sesam_rollback (PHP 3 CVS only)

Verwirft alle noch nicht gesicherten Änderungen der SESAM-Datenbank

bool **sesam_rollback** (void)

Rückgabewerte: `TRUE` bei erfolgreicher Ausführung, ansonsten `FALSE`

sesam_rollback() Verwirft alle noch nicht gesicherten Änderungen der SESAM-Datenbank sowie Ergebnis-Cursor und Ergebnis-Deskriptoren.

Bei jeder Beendigung eines aktuellen Skripts sowie beim Aufruf von sesam_disconnect()) werden nicht gesicherte Daten durch einen impliziten **sesam_rollback()** Aufruf verworfen.

Siehe auch: sesam_commit().

Beispiel 1. Verwerfen einer Änderung der SESAM-Datenbank

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
  if (sesam_execimm("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
   && sesam_execimm("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
     sesam_commit();
  else
     sesam_rollback();
}
?>
```

sesam_execimm (PHP 3 CVS only)

Ausführen einer "immediate" SQL-Anweisung

string **sesam_execimm** (string query)

Rückgabewerte: Ein SESAM "result identifier" oder `FALSE` im Fehlerfall.

sesam_execimm() bewirkt die Ausführung einer "immediate" SQL-Anweisung wie z.B. UPDATE, INSERT oder DELETE, die keine Rückgabewerte liefern und weder Eingabe- noch Ausgabevariablen verwenden und setzt den Abfrageparameter *`affected_rows`* über die sesam_affected_rows() Funktion. Auf "select type" Abfragen kann **sesam_execimm()** nicht angewendet werden.

Hinweis: sesam_query() unterstützt sowohl "immediate" als auch "select-type" Abfragen. **sesam_execimm()** sollte nur verwendet werden, wenn der verwendete Anfragetyp von vornherein bekannt ist. SELECT-Abfragen mit **sesam_execimm()** werden mit dem Fehler `$err["sqlstate"] == "42SBW"` beendet.

Der Rückgabewert "result identifier" kann nicht für Abfragen mit sesam_affected_rows() verwendet werden; er wird wegen der Kompatibilität zur Funktion sesam_query() ausgegeben.

```
$stmt = "INSERT INTO mytable VALUES('one', 'two')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print("sqlstate = ".$err["sqlstate"]."\n".
      "Affected rows = ".$err["rowcount"]." == ".
      sesam_affected_rows($result)."\n");
```

Siehe auch: sesam_query() und sesam_affected_rows().

sesam_query (PHP 3 CVS only)

Durchführen einer SESAM-SQL-Abfrage und aufbereiten der Ergebnisse

string **sesam_query** (string query [, bool scrollable])

Rückgabewerte: ein SESAM "result identifier" oder `FALSE` im Fehlerfall.

Von anderen Funktionen wird "result_id" für Abfrage-Ergebnisse verwendet.

sesam_query() schickt eine Abfrage an die aktuelle Datenbank auf dem Server. Sie kann sowohl "immediate" SQL-Anweisungen als auch "select type" Abfragen ausführen. Bei der Ausführung einer "immediate" Anweisung ist kein Cursor aktiv und bei jedem folgenden sesam_fetch_row() oder sesam_fetch_result() call will return an empty Aufruf wird ein leeres Ergebnis (zero columns, indicating end-of-result) zurückgeliefert. Für "select type" Anweisungen wird ein Ergebnis-Deskriptor und ein Cursor (scrollable oder sequential, abhängig vom optionalen bool *`scrollable`*-Parameter) zugewiesen. Wenn *`scrollable`* nicht angegeben wird, ist der Cursor sequentiell.

Bei der Verwendung von "scrollable" Cursors kann der Cursor im Ausgabesatz beliebig positioniert werden. Für jede "scrollable" Abfrage gibt es globale Standardwerte für den "scrolling"-Typ (Initial-Wert: `SESAM_SEEK_NEXT`) und den scrolling offset, der entweder einmal durch sesam_seek_row() oder bei jedem Lesen einer Zeile mit sesam_fetch_row().

Für "immediate" Befehle wird die Anzahl der betroffenen Zeilen für Abfragen von der Funktion sesam_affected_rows() gesichert.

Siehe auch: sesam_fetch_row() und sesam_fetch_result().

Beispiel 1. Zeige alle Zeilen der Tabelle "Telefon" als HTML-Tabelle

```
<?php
if (!sesam_connect("phonedb", "demo", "otto"))
  die("cannot connect");
$result = sesam_query("select * from phone");
if (!$result) {
  $err = sesam_diagnostic();
  die($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array($result)) {
  echo " <TR><TH COLSPAN=".$cols["count"]."">Result:</TH></TR>\n";
  echo " <TR>\n";
  for ($col = 0; $col < $cols["count"]; ++$col) {
    $colattr = $cols[$col];
    /* Span the table head over SESAM's "Multiple Fields": */
    if ($colattr["count"] > 1) {
      echo "  <TH COLSPAN=".$colattr["count"]."">".$colattr["name"].
           "(1..".$colattr["count"].")</TH>\n";
      $col += $colattr["count"] - 1;
    }
    else
      echo "  <TH>" . $colattr["name"] . "</TH>\n";
  }
  echo " </TR>\n";
}
```

```
do {
  // Fetch the result in chunks of 100 rows max.
  $ok = sesam_fetch_result($result,100);
  for ($row=0; $row < $ok["rows"]; ++$row) {
    echo " <TR>\n";
    for ($col = 0; $col < $ok["cols"]; ++$col) {
      if (isset($ok[$col][$row]))
        echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
      else
        echo "  <TD>-empty-</TD>\n";
    }
    echo " </TR>\n";
  }
} while ($ok["truncated"]); // while there may be more data
echo "</TABLE>\n";
// free result id
sesam_free_result($result);
?>
```

sesam_num_fields (PHP 3 CVS only)

Liefert die Anzahl von Feldern/Spalten in einem Ausgabesatz

int **sesam_num_fields** (string result_id)

Nach dem Aufruf der Funktion sesam_query() mit einer "select type" Abfrage wird die Anzahl von Spalten als Resultat ausgegeben. Die Ausgabe ist ein Integer-Wert, der die Anzahl aller Spalten (aka.fields) im aktuellen *result_id* Ergebnis-Satz oder `FALSE`.

Bei "immediate" Befehle wird der Wert `NULL` zurückgegeben. Die SESAM "multiple field" Spalten zählen entsprechend der jeweiligen Dimension, d.h. ein dreispaltiges "multiple field" zählt als drei Spalten.

Siehe auch: sesam_query() und sesam_field_array() zur Unterscheidung zwischen "multiple field" Spalten und normalen Spalten.

sesam_field_name (PHP 3 CVS only)

Liefert einen Spaltennamen des Ausgabesatzes

int **sesam_field_name** (string result_id, int index)

Liefert den Namen eines Feldes (d.h. den Spaltennamen) oder `FALSE` on error.

Bei "immediate" Abfragen oder bei dynamischen Spalten wird ein Leerstring zurückgegeben.

Anmerkung: Der Spalten-Index zählt ab `NULL`, nicht wie bei SESAM ab Eins.

Siehe auch: sesam_field_array(). Bietet einen bequemeren Zugriff auf Spaltennamen und -typen und erlaubt die Prüfung auf "multiple fields".

sesam_diagnostic (PHP 3 CVS only)

Liefert Status-Informationen über den letzten SESAM-Aufruf

array **sesam_diagnostic** (void)

Liefert ein assoziatives Array (???) von Status- und Returncodes des letzten SQL- Aktion (Abfrage/Statement(???)/Befehl). Elemente des Arrays sind:

Tabelle 1. Von sesam_diagnostic() gelieferte Status Information

Element	Contents
$array["sqlstate"]	5-stelliger SQL-Returncode (mögliche Werte von SQLSTATE: siehe SESAM-Benutzerhandbuch)
$array["rowcount"]	Anzahl betroffener Zeilen im letzten update/insert/delete (nur nach "immediate" Statements gesetzt)
$array["errmsg"]	"lesbarer" Fehlermeldungstext (nur nach Fehlern gesetzt)
$array["errcol"]	Spaltennummer des letzten Fehlers (1-indiziert oder 0, wenn nicht definiert; nur nach Fehlern gesetzt)
$array["errlin"]	Zeilennummer des letzten Fehlers (1-indiziert oder 0, wenn nicht definiert; nur nach Fehlern gesetzt)

Im folgenden Beispiel wird ein Syntaxfehler (E SEW42AE ILLEGAL CHARACTER) zusammen mit dem fehlerhaften SQL-Statement und der genauen Position des Fehlers angezeigt.

Beispiel 1. Anzeige von SESAM-Fehlermeldungen mit Fehlerposition

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode($exec_str)
{
   $err = Sesam_Diagnostic();
   $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
   if ($err["errlin"] == -1)
     --$colspan;
   if ($err["errcol"] == -1)
     --$colspan;
   if ($err["rowcount"] == 0)
     --$colspan;
   echo "<TABLE BORDER>\n";
   echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
```

```
    htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
      echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
      $errstmt = $exec_str."\n";
      for ($lin=0; $errstmt != ""; ++$lin) {
        if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
          if (! ($i = strchr($errstmt, "\n")))
            $i = "";
          $line = substr($errstmt, 0, strlen($errstmt)-strlen($i)+1);
          $errstmt = substr($i, 1);
          if ($line != "\n")
            print htmlspecialchars($line);
        }
        else {
          if (! ($i = strchr($errstmt, "\n")))
            $i = "";
          $line = substr($errstmt, 0, strlen($errstmt)-strlen($i)+1);
          $errstmt = substr($i, 1);
          for ($col=0; $col < $err["errcol"]; ++$col)
            echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
          echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
          print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
          for ($col=0; $col < $err["errcol"]; ++$col)
            echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
          echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
        }
      }
      echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
      echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
      echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
      echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect("mycatalog", "phoneno", "otto"))
  die("cannot connect");

$stmt = "SELECT * FROM phone\n".
        "  WHERE@ LASTNAME='KRAEMER'\n".
        "  ORDER BY FIRSTNAME";
if (! ($result = sesam_query($stmt)))
  PrintReturncode($stmt);
?>
```

Siehe auch: sesam_errormsg() für einfachen Zugriff nur auf die Fehlermeldung

sesam_fetch_result (PHP 3 CVS only)

Liefert das Ergebnis einer Abfrage (ganz oder teilweise)

mixed **sesam_fetch_result** (string result_id [, int max_rows])

Liefert ein gemischtes Array (???) mit den Ergebnissen einer Abfrage, optional begrenzt auf ein Maximum durch *max_rows* Zeilen. Sowohl Zeilen- als auch Spaltenindizes sind Null-basiert.

Tabelle 1. Gemischtes Resultat mit der Funktion sesam_fetch_result()

Array Element	Contents
int $arr["count"]	Anzahl von Ergebnis-Spalten im Ausgabesatz (oder `NULL` bei "immediate" Abfragen)
int $arr["rows"]	Anzahl von Zeilen im Ausgabesatz (zwischen `NULL` und *max_rows*)
bool $arr["truncated"]	`TRUE` wenn die Anzahl der Zeilen mindestens *max_rows* ist, ansonsten `FALSE`. Hinweis: auch bei `TRUE` kann der nächste **sesam_fetch_result()** u.U. 0 Zeilen zurückliefern, weil es keine weiteren Ergebniseinträge gibt.
mixed $arr[col][row]	Ergebnisdaten für alle Felder in Zeile(`row`) und Spalte(`col`), wobei der Integer-Index `row` zwischen 0 and `$arr["rows"]-1`, und `col` zwischen 0 und `$arr["count"]-1`) liegt. Da Felder auch leer sein können, muß ihre Existenz mit der PHP-Funktion isset() verifiziert werden. Der zurückgelieferte Feldtyp hängt vom entsprechenden, für die Spalte defineirten SQL-Typ ab (siehe SESAM overview für die angewendeten Umsetzungen). SESAM "multiple fields" sind "inlined" (???) und werden wie eine Spaltenfolge behandelt.

Hinweis: der Speicherbedarf für eine umfangreiche Abfrage kann immens groß werden. Es ist empfehlenswert, den Parameter *max_rows* zum Begrenzen der maximalen Anzahl von gelieferten Zeilen zu verwenden, um zu verhindern, daß die Ausgabe nicht den gesamten verfügbaren Speicher belegt.

Siehe auch: sesam_fetch_row(), und sesam_field_array() zur Überprüfung auf "multiple fields". Ein komplettes Beispiel für die Funktion **sesam_fetch_result()** findet man in sesam_query().

sesam_affected_rows (PHP 3 CVS only)

Anzahl der von einer sofortigen (???) Abfrage betroffenen Zeilen

int **sesam_affected_rows** (string result_id)

result_id ist eine von sesam_query() zurückgelieferte gültige Ergebniskennung (???).

Liefert die Anzahl der von einer Abfrage betroffenen Zeilen bezogen auf die *result_id*.

Die Funktion **sesam_affected_rows()** liefert sinnvolle Werte nur in Verbindung mit "immediate" SQL-Statements (Update-Operationen wie `INSERT`, `UPDATE` and `DELETE`), da SESAM für "select type" Abfragen keinerlei Information über betroffene Zeilen liefert. Der gelieferte Wert ist die Anzahl der betroffenen Zeilen.

Siehe auch: sesam_query() und sesam_execimm()

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper($name)."'");
if (! $result) {
    ... error ...
}
print sesam_affected_rows($result).
      " entries with last name ".$name." deleted.\n"
```

sesam_errormsg (PHP 3 CVS only)

Liefert die Fehlermeldung des letzten SESAM-Aufrufs

string **sesam_errormsg** (void)

Liefert die SESAM-Fehlermeldung des letzten SESAM-Fehlers.

```
if (!sesam_execimm($stmt))
  printf("%s<br>\n", sesam_errormsg());
```

Siehe auch: sesam_diagnostic() für den vollen Umfang der SESAM SQL Statusinformation.

sesam_field_array (PHP 3 CVS only)

Liefert Meta-Information über einzelne Ergebnisspalten

array **sesam_field_array** (string result_id)

`result_id` ist eine gültige Ergebniskennung (?) der Funktion sesam_query().

Liefert eine gemischtes assoziativ indiziertes Array (???) mit Meta-Informationen (Spaltenname, Typ, Genauigkeit, ...) über einzelne Ergebnisspalten nach einer mit *`result_id`* assoziierten Abfrage.

Tabelle 1. Von sesam_field_array() gelieferter gemischter Ausgabesatz

Array Element	Contents
int $arr["count"]	Gesamtanzahl von Zeilen im Ausgabesatz (oder `NULL` bei "immediate" Abfragen). SESAM "multiple fields" sind "inlined" (???) und werden werden wie die entsprechende Anzahl von Spalten behandelt.

Array Element	Contents
string $arr[col]["name"]	Spaltenname für column(`col`), wobei `col` zwischen 0 und `$arr["count"]-1` liegt. Für dynamisch berechnete Spalten kann der gelieferte Wert auch ein Leer-String sein. SESAM "multiple fields" sind "inlined" (???) und werden wie die entsprechende Anzahl von Spalten behandelt , jedes mit dem selben Spaltennamen.
string $arr[col]["count"]	"count" ist der Wiederholungsfaktor, wenn die Spalte als "multiple field" definiert wurde (normalerweise 1). Die erste Spalte einer "multiple field"-Spalte enthält dagegen die Anzahl von Wiederholungen (in der zweiten und in allen weiteren Spalten ist "count" 1), was zur Identifikation von "multiple fields" im Ergebnissatz verwendet werden kann. Das Beispiel in der Beschreibung von sesam_query() demonstriert die Anwendung des "count"-Attributes.
string $arr[col]["type"]	PHP-Variablentyp der Daten für column(`col`), wobei `col` zwischen und `$arr["count"]-1` liegt. Der gelieferte Wert kann (abhängig vom SQL-Typ des Ergebnisses) einer der folgenden sein: • "integer" • "double" • "string" SESAM "multiple fields" sind "inlined" (???) und werden wie die entsprechende Anzahl von Spalten behandelt, jedes mit dem selben PHP-Typ.

Array Element	Contents
string $arr[col]["sqltype"]	SQL-Variablentyp der Spaltendaten für column(`col`), wobei `col` zwischen 0 und `$arr["count"]-1` liegt. Der gelieferte Wert kann abhängig vom SQL-Typ des Ergebnisses einer der folgenden sein: • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" SESAM "multiple fields" sind "inlined" (???) und werden wie die entsprechende Anzahl von Spalten behandelt, jedes mit dem selben SQL-Typ.
string $arr[col]["length"]	Das Attribut "length" der SQL-Variablen in column(`col`), wobei `col` zwischen 0 und `$arr["count"]-1` liegt. Das "length" Attribut wird in "CHARACTER" und "VARCHAR" SQL-Typen zur Spezifizierung der (maximalen) Länge der Zeichenvariablen verwendet. SESAM "multiple fields" sind "inlined" (???) und werden wie die entsprechende Anzahl von Spalten behandelt, jedes mit dem selben Längenattribut.
string $arr[col]["precision"]	Das Attribut "precision" der SQL-Variablen in column(`col`), wobei `col` zwischen 0 and `$arr["count"]-1` liegt. Das Attribut "precision" wird für numerische und Zeit-Felder verwendet. SESAM "multiple fields" sind "inlined" (???) und werden wie die entsprechende Anzahl von Spalten behandelt, jedes mit dem selben Attribut "precision".

Array Element	Contents
string $arr[col]["scale"]	Das Attribut "scale" der SQL-Variablen in column(`col`), wobei `col` zwischen 0 and `$arr["count"]-1` liegt. Das Attribut "scale" wird für numerische Datentypen verwendet. SESAM "multiple fields" sind "inlined" (???) und werden wie die entsprechende Anzahl von Spalten behandelt, jedes mit dem selben Attribut "scale".

Siehe Beschreibung von sesam_query() für eine beispielhafte Anwendung der Funktion **sesam_field_array()**.

sesam_fetch_row (PHP 3 CVS only)

Lesen einer Zeile als Array

array **sesam_fetch_row** (string result_id [, int whence [, int offset]])

Liefert ein Array entsprechend der gelesenen Zeile oder `FALSE` nach de letzten gefundenen Zeile.

Die Anzahl der Spalten im Ausgabesatz wird als Element des assoziativen Arrays $array["count"] geliefert. Da Ergebnisspalten auch leer sein können, kann die Funktion count() nicht auf das Ergebnis der Funktion **sesam_fetch_row()** angewendet werden.

`result_id` ist eine gültige Ergebniskennung (???) der Funktion sesam_query() (nur Abfragen vom Typ select!).

`whence` ist ein optionaler Parameter für eine "fetch"-Operation (???) bei "scrollable" Cursor und kann auf folgende vordefinierte Werte gesetzt werden:

Tabelle 1. Gültige Werte für Parameter *`"whence"`*

Value	Constant	Meaning
0	`SESAM_SEEK_NEXT`	sequentielles Lesen (nach "fetch" wird der interne Standardwert auf `SESAM_SEEK_NEXT` gesetzt)
1	`SESAM_SEEK_PRIOR`	sequentielles Lesen rückwärts (nach "fetch" wird der interne Standardwert auf `SESAM_SEEK_PRIOR` gesetzt)
2	`SESAM_SEEK_FIRST`	zurück zur ersten Zeile (nach "fetch" wird der interne Standardwert auf `SESAM_SEEK_NEXT` gesetzt)
3	`SESAM_SEEK_LAST`	suche letzte Zeile (nach "fetch" wird der interne Standardwert auf `SESAM_SEEK_PRIOR` gesetzt)

Value	Constant	Meaning
4	SESAM_SEEK_ABSOLUTE	suche Zeilen mit der Nummer definiert durch *offset* (Null-basierte Indizierung (nach "fetch" wird der interne Standardwert auf SESAM_SEEK_ABSOLUTE gesetzt und der interne offset-Wert wird automatisch hochgesetzt)
5	SESAM_SEEK_RELATIVE	suche relativ zur aktuellen Position; *offset* kann positiv oder negativ sein.

Dieser Parameter ist nur für "scrollable" Cursors gültig.

Bei "scrollable" kann der Cursor beliebig im Ausgabesatz positioniert werden. Wird der optionale Parameter *whence* nicht benutzt, werden die globalen Standardwerte für den Scroll-Typ verwendet (Initialiwert: SESAM_SEEK_NEXT, einstellbar mit der Funktion sesam_seek_row()). Bei Verwendung des Parameters *whence* ersetzen dessen Werte die golbalen Standardwerte.

offset ist ein optionaler Parameter, der nur wirkt (und auch nur benötigt wird), wenn *whence* entweder den Wert SESAM_SEEK_RELATIVE oder SESAM_SEEK_ABSOLUTE hat. Dieser Parameter ist nur für "scrollable" Cursors gültig.

sesam_fetch_row() holt eine Datenzeile aus dem Ergebnis mit der zugeordneten Ergebniskennung (???). Die Zeile wird als Array geliefert, wobei die indizierten Werte zwischen 0 und $array["count"]-1) liegen. Da Felder auch leer sein können, muß die Existenz eines mit der PHP-Funktion isset() verifiziert werden. Der Typ der gelieferten Felder ist abhängig von der entsprechenden SQL-Typdeklaration düe die Spalten (siehe SESAM overview für die angewendeten Umsetzungen). SESAM "multiple fields" sind "inlined" (???) und werden wie eine Spaltenfolge behandelt.

Nachfolgende Aufrufe von **sesam_fetch_row()** liefern die nächste Zeile (oder die vorhergehende, höchste/niedrigste je nach den Attributen) im Ausgabesatz oder FALSE wenn keine Zeilen mehr verfügabr sind.

Beispiel 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (! $result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array($row)) {
    print " <TR>\n";
    for($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
```

```
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

Siehe auch: Funktion sesam_fetch_array() (liefert ein assoziatives Array) und sesam_fetch_result() (liefert viele Zeilen pro Aufruf (???))

sesam_fetch_array (PHP 3 CVS only)

Liefert eine Zeile als assoziatives Array

array **sesam_fetch_array** (string result_id [, int whence [, int offset]])

Liefert ein Array, das der gelesenen Zeile entspricht oder `FALSE` falls es keine weiteren Zeilen mehr gibt.

sesam_fetch_array() ist eine Alternative zur Funktion sesam_fetch_row(). Die Daten werden anstatt in numerischen Indizes des Ergebnisarrays in assoziativen Indizes gespeichert, wobei die Feldnamen als Indizes benutzt werden.

`result_id` ist eine gültige Ergebniskennung (???) der Funktion sesam_query() (nur Abfragen vom Typ select!).

Für die gültigen Werte der optionalen Parameter *`whence`* und *`offset`* im Detail siehe sesam_fetch_row().

Die Funktion **sesam_fetch_array()** holt eine mit der spezifizierten Ergebniskennung (???) assoziierte Datenzeile. Die Zeile wird als assoziatives Array zur Verfügung gestellt. Jede Ergebnisspalte wird mit einem assoziativen Index gleich dem Spaltennamen (aka. field) abgespeichert. Die Spaltennamen werden in Kleinschreibung konvertiert dargestellt.

Spalten ohne Feldnamen (z.B. Ergebnisse von arithmetischen Operationen) und leere Felder werden im Array nicht gespeichert. Haben zwei oder mehr Spalten im Ergebnis den selben Namen, bekommt diesen die letzte Spalte. In diesem Fall empfiehlt sich der Einsatz von sesam_fetch_row() oder Alias-Definitionen für die Spalten.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

Eine Spezialbehandlung erlaubt den Zugriff auf "multiple field" Spalten (die sonst alle den selben Spaltennamen haben würden). Für jede Spalte eines "multiple field" wird der Name gebildet durch Anhängen von "(n)", wobei n den Sub-Index der "multiple field" Spalte darstellt und Werte von 1 bis zum deklarierten Wiederholungsfaktor annehmen kann. Zur Übereinstimmung mit der entsprechenden Abfragesyntax sind die Indizes n i c h t null-basiert. Für eine Spalte der Definition:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

ergäben sich die assoziativen Indizes für die einzelnen "multiple fields" als `"multi(1)"`, `"multi(2)"`, and `"multi(3)"`.

Nachfolgende Aufrufe von **sesam_fetch_array()** liefern die nächste Zeile (oder die vorhergehende, höchste/niedrigste je nach den Attributen) im Ausgabesatz oder `FALSE` wenn keine Zeilen mehr verfügabr sind.

Beispiel 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (! $result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count($row) > 0) {
    print " <TR>\n";
    print "  <TD>".htmlspecialchars($row["firstname"])."</TD>\n";
    print "  <TD>".htmlspecialchars($row["lastname"])."</TD>\n";
    print "  <TD>".htmlspecialchars($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

Siehe auch: sesam_fetch_row() (liefert ein indiziertes Array)

sesam_seek_row (PHP 3 CVS only)

Setzt "scrollable Cursor" Modus für weitere Aufrufe

bool **sesam_seek_row** (string result_id, int whence [, int offset])

`result_id` ist eine gültige Ergebniskennung (???); nur bei Abfragen vom Typ select und nur, wenn beim Aufruf von sesam_query()) ein "scrollable Cursor" verlangt wurde.

`whence` definiert den globalen Standardwert für den Scroll-Typ, der in weiteren Lese-Operationen für "scrollable" Cursor angewendet wird; folgende vordefinierte Werte sind möglich: constants:

Tabelle 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	`SESAM_SEEK_NEXT`	sequentielles Lesen
1	`SESAM_SEEK_PRIOR`	sequentielles Lesen rückwärts
2	`SESAM_SEEK_FIRST`	erste Zeile lesen (nach dem Lesen wird der Standardwert auf `SESAM_SEEK_NEXT` gesetzt)
3	`SESAM_SEEK_LAST`	letzte Zeile lesen (nach dem Lesen wird der Standardwert auf `SESAM_SEEK_PRIOR` gesetzt)

Value	Constant	Meaning
4	SESAM_SEEK_ABSOLUTE	lesen Zeile mit der Nummer definiert durch *offset* (Null-basierte Indizierung; nach "fetch" wird der interne Standardwert auf SESAM_SEEK_ABSOLUTE gesetzt und der interne offset-Wert wird automatisch hochgesetzt)
5	SESAM_SEEK_RELATIVE	suche relativ zur aktuellen Position; *offset* kann positiv oder negativ sein und wird auch als Standard-Offset für weitere Abfragen gesetzt

offset ist ein optionaler Parameter, der nur wirkt (und auch nur benötigt wird), wenn *whence* entweder den Wert SESAM_SEEK_RELATIVE oder SESAM_SEEK_ABSOLUTE hat.

sesam_free_result (PHP 3 CVS only)

Gibt Betriebsmittel für die Abfrage frei (der Abfrage ???)

int **sesam_free_result** (string result_id)

Gibt Betriebsmittel der mit *result_id* assoziierten Abfrage frei. Rückgabewert im Fehlerfall: FALSE.

XC. Session Funktionen

Die Unterstützung von Sessions in PHP bietet die Möglichkeit, bestimmte Daten während einer Folge von Aufrufen ihrer Website festzuhalten. Damit können Sie persönlichere Applikationen erstellen und Ihre Website ansprechender gestalten.

Falls sie mit dem Sessionmanagment von PHPLIB vertraut sind, werden sie Ähnlichkeiten zu PHP's Session-Unterstützung feststellen.

Einem Besucher wird beim Aufruf ihrer Website eine einzigartige ID, die sogenannte Session-ID, zugeordnet. Diese wird entweder benutzerseitig in einem Cookie abgelegt oder in der URL übermittelt.

Die Unterstützung von Sessions erlaubt Ihnen, eine beliebige Anzahl von Variablen zu registrieren und diese über Anfragen hinweg zu erhalten. Wenn ein Besucher Ihre Website aufruft, überprüft PHP automatisch (falls session.auto_start auf 1 gesetzt ist) oder auf Anfrage (explizit durch session_start() oder implizit durch session_register()), ob mit der Anfrage schon eine bestimmte Session-ID gesendet wurde. Wenn dies der Fall ist, wird die zuvor gespeicherte Umgebung wiederhergestellt.

Alle registrierten Variablen werden serialisiert, nachdem die Anfrage beendet ist. Registrierte Variablen, die nicht definiert sind, werden als nicht definiert gekennzeichnet. Auch bei darauffolgenden Zugriffen werden sie nicht vom Session-Modul definiert, außer der Benutzer definiert sie später.

Die Konfigurationseinstellungen von `track_vars` und `register_globals` beeinflussen, wie die Session-Variablen gespeichert und wiederhergestellt werden.

> **Anmerkung:** Seit PHP 4.0.3 ist `track_vars` immer aktiviert.

> **Anmerkung:** Seit PHP 4.1.0 steht `$_SESSION` genau wie `$_POST`, `$_GET`, `$_REQUEST` und so weiter, als globale Variable zur Verfügung. Im Gegensatz zu `$HTTP_SESSION_VARS` ist `$_SESSION` immer global. Deshalb sollte `global` nicht für `$_SESSION` verwendet werden.

Falls `track_vars` aktiviert und `register_globals` deaktiviert ist, können nur Bestandteile des globalen assoziativen Arrays `$HTTP_SESSION_VARS` als Session-Variablen registriert werden. Wiederhergestellte Session-Variablen stehen nur im Array `$HTTP_SESSION_VARS` zur Verfügung.

Beispiel 1. Registrierung einer Variablen bei aktiviertem `track_vars`

```
<?php
if (isset($HTTP_SESSION_VARS['zaehler'])) {
   $HTTP_SESSION_VARS['zaehler']++;
}
else {
   $HTTP_SESSION_VARS['zaehler'] = 0;
}
?>
```

Aus Gründen der Sicherheit und der Lesbarkeit des Codes wird die Verwendung von `$_SESSION` (oder `$HTTP_SESSION_VARS` bei PHP 4.0.6 oder niedriger) empfohlen. Mit `$_SESSION` oder `$HTTP_SESSION_VARS` werden die Funktionen session_register()/session_unregister()/session_is_registered() nicht benötigt. Der Benutzer kann auf die Session-Variable wie auf eine normale Variable zugreifen.

Beispiel 2. Registrierung einer Variablen mit $_SESSION.

```
<?php
// Verwenden Sie bei PHP 4.0.6 oder niedriger $HTTP_SESSION_VARS
if (!isset($_SESSION['zaehler'])) {
    $_SESSION['zaehler'] = 0;
}
else {
    $_SESSION['zarhler']++;
?>
```

Beispiel 3. Aufheben der Registrierung einer Variablen mit $_SESSION.

```
<?php
// Verwenden Sie bei PHP 4.0.6 oder niedriger $HTTP_SESSION_VARS
unset($_SESSION['zaehler']);
?>
```

Wenn `register_globals` aktiviert ist, können alle globalen Variablen als Session-Variablen registriert werden und die Session-Variablen werden als entsprechende globale Variablen wiederhergestellt. Da PHP wissen muss, welche globalen Variablen als Session-Variablen registriert sind, muss der Benutzer Variablen mit der Funktion session_register() registrieren, während

$HTTP_SESSION_VARS/$_SESSION die Verwendung von session_register() nicht benötigt.

Achtung

Wenn Sie $HTTP_SESSION_VARS/$_SESSION verwenden und register_globals deaktivieren, sollten Sie session_register(), session_is_registered() und session_unregister() nicht verwenden.

Wenn sie register_globals aktivieren, sollten Sie session_unregister() verwenden, weil beim Deserialisieren von Session-Daten die Session-Variablen als globale Variablen registriert werden. Die Deaktivierung von register_globals ist sowohl aus Sicherheitsgründen als auch wegen der Performance empfehlenswert.

Beispiel 4. Registrierung einer Variablen bei aktiviertem register_globals

```
<?php
if (!session_is_registered('zaehler')) {
    session_register("zaehler");
    $zaehler = 0;
}
else {
    $zaehler++;
}
?>
```

Wenn sowohl track_vars als auch register_globals aktiviert sind, dann referenzieren die globalen Variablen und die Einträge von $HTTP_SESSION_VARS/$_SESSION für bereits registrierte Variablen den selben Wert.

Wenn der Benutzer session_register() verwendet, um eine Session-Variable zu registrieren, befindet sich diese Variable nicht vor dem nächsten Laden aus dem Sessionspeicher (d.h. bei der nächsten Anfrage) im Array von $HTTP_SESSION_VARS/$_SESSION.

Es gibt zwei Methoden, eine Session-ID zu übermitteln:

- Cookies
- URL Parameter

Das Session-Modul unterstützt beide Methoden. Cookies sind optimal, aber da sie nicht zuverlässig sind (Clients müssen sie nicht unbedingt akzeptieren), können wir uns nicht auf sie verlassen. Die zweite Methode hängt die Session-ID direkt an die URLs.

Wenn PHP mit --enable-trans-sid kompiliert wurde, kann es dies transparent machen. Wenn Sie diese Option aktivieren, werden relative URIs automatisch geändert, sodass sie die Session-ID enthalten. Alternativ können sie die Konstante SID verwenden, die definiert wird, falls vom Client

keinen passendes Cookie gesendet wurde. `SID` hat entweder die Form `session_name=session_id` oder ist eine leere Zeichenkette.

Das folgende Beispiel demonstriert, wie eine Variable registriert wird und wie unter Verwendung einer SID korrekt auf eine andere Seite verwiesen wird.

Beispiel 5. Zählen der Seitenaufrufe eines einzelnen Benutzers

```
<?php
if (!session_is_registered('zaehler')) {
    session_register('zaehler');
    $zaehler = 1;
}
else {
    $zaehler++;
}
?>

Hallo Besucher, Sie haben diese Seite <?php echo $zaehler; ?> Mal
aufgerufen.<p>

<?php
# für den Fall, dass der Benutzer Cookies deaktiviert hat,
# ist das <?php echo SID?> nötig, um die Session-ID zu erhalten
# (wenn die Option short tag aktiviert ist, kann <?=SID?> verwendet
# werden)
?>

Hier gehts <A HREF="nextpage.php?<?php echo SID?>">weiter</A>
```

Das `<?=SID?>` ist nicht nötig, wenn PHP mit `--enable-trans-sid` kompiliert wurde.

> **Anmerkung:** Bei nichtrelativen URLs wird davon ausgegangen, dass sie auf externe Seiten zeigen und, weil es ein Sicherheitsrisiko wäre, die SID an einen anderen Server zu übermitteln, von daher keine SID angehängt.

Wenn Sie die Speicherung in einer Datenbank oder irgendeine andere Art der Speicherung realisieren wollen, müssen Sie session_set_save_handler() verwenden, um eine Reihe von benutzerdefinierten Speicherfunktionen zu erzeugen.

Das Sessionmanagementsystem unterstützt eine Anzahl von Konfigurationsoptionen, die sie in Ihrer php.ini setzen können. Wir geben Ihnen dazu einen kleinen Überblick.

- `session.save_handler` definiert den Namen der Prozedur, die benutzt wird, um die Daten zu speichern und zurückzuholen, die mit der Session in Verbindung stehen. Grundeinstellung `files`.
- `session.save_path` definiert das Argument, das an die Speicherprozedur übergeben wird. Wenn Sie die standardmäßige files Prozedur wählen, ist das der Pfad, unter dem die Dateien

erzeugt werden. Grundeinstellung `/tmp`. Wenn die Verzeichnistiefe von `session.save_path` größer als zwei ist, wird die garbage collection nicht durchgeführt.

> **Warnung**
>
> Wenn Sie ein Verzeichnis gewählt haben, für das jeder Leserechte hat, wie das z.B. bei /tmp (Grundeinstellung) der Fall ist, könnten andere Serverbenutzer mit Hilfe der Dateiliste dieses Verzeichnisses Ihre Sessions entführen.

- `session.name` spezifiziert den Namen der Session, der als Cookie-Name verwendet wird. Grundeinstellung `PHPSESSID`.
- `session.auto_start` spezifiziert, ob das Session-Modul zu Beginn einer Anfrage automatisch eine Session startet. Grundeinstellung `0` (deaktiviert).
- `session.cookie_lifetime` spezifiziert die Cookie-Lebensdauer, die an den Browser geschickt wird, in Sekunden. Der Wert 0 bedeutet "bis der Browser geschlossen wird." Grundeinstellung `0`.
- `session.serialize_handler` definiert den Namen der Prozedur, die benutzt wird, um Daten zu serialisieren/deserialisieren. Gegenwärtig wird ein internes PHP-Format (Name `php`) und WDDX (name `wddx`) unterstützt. WDDX steht nur zur Verfügung, wenn PHP mit WDDX support kompiliert wurde. Grundeinstellung `php`.
- `session.gc_probability` spezifiziert die prozentuale Wahrscheinlichkeit, dass die gc (garbage collection) Routine bei jeder Anfrage gestartet wird. Grundeinstellung `1`.
- `session.gc_maxlifetime` spezifiziert die Anzahl der Sekunden, nach denen Daten als 'garbage' ('Müll') betrachtet und entsorgt werden.
- `session.referer_check` enthält die Zeichenkette, auf die Sie jeden HTTP-Referer überprüfen wollen. Wenn der Referer vom Client gesendet und die Zeichenkette nicht gefunden wurde, wird die eingebettete Session-ID als ungültig gekennzeichnet. Grundeinstellung ist eine leere Zeichenkette.
- `session.entropy_file` gibt den Pfad zu einer externen Quelle (Datei) an, die bei der Erzeugung einer Session-ID als zusätzliche Entropiequelle verwendet wird. Beispiele sind `/dev/random` oder `/dev/urandom`, die auf vielen Unix-Systemen zur Verfügung stehen.
- `session.entropy_length` spezifiziert die Anzahl der Bytes, die von der oben spezifizierten Datei gelesen werden. Grundeinstellung `0` (deaktiviert).
- `session.use_cookies` spezifiziert, ob das Modul Cookies verwendet, um die Session-ID clientseitig zu speichern. Grundeinstellung `1` (aktiviert).
- `session.cookie_path` spezifiziert den Pfad, in dem das Session-Cookie gesetzt wird. Grundeinstellung `/`.
- `session.cookie_domain` spezifiziert die Domain, unter der das Session-Cookie gesetzt wird. In der Grundeinstellung überhaupt keine.
- `session.cache_limiter` spezifiziert die Methode der Cacheverwaltung, die bei Session-Seiten benutzt wird (none/nocache/private/private_no_expire/public). Grundeinstellung

`nocache`.

- `session.cache_expire` spezifiziert in Minuten, wie lange Session-Seiten im Cache bleiben. Bei nocache ist diese Angabe wirkungslos. Grundeinstellung `180`.
- `session.use_trans_sid` bestimmt, wenn mit `--enable-trans-sid` kompiliert wurde, ob transparente SID-Unterstützung aktiviert ist oder nicht. Grundeinstellung `1` (aktiviert).
- `url_rewriter.tags` bestimmt, wenn Unterstützung für transparente SID aktiviert ist, welche HTML-Tags so umgeschrieben werden, dass sie die Session-ID beinhalten. Grundeinstellung `a=href,area=href,frame=src,input=src,form=fakeentry`

Anmerkung: Die Behandlung von Sessions wurde in PHP 4.0 eingeführt.

session_start (PHP 4 >= 4.0.0)

Initialisiert eine Session

bool **session_start** (void)

session_start() erzeugt eine Session (oder nimmt die aktuelle, basierend auf der Session-ID, die über eine GET-Variable oder ein Cookie übermittelt wurde, wieder auf).

Wenn Sie eine benannte Session verwenden wollen, müssen Sie session_name() aufrufen, bevor Sie **session_start()** aufrufen.

Diese Funktion gibt immer `TRUE` zurück.

> **Anmerkung:** Bei Verwendung Cookie-basierter Sessions müssen Sie **session_start()** aufrufen, bevor irgend etwas an den Browser geschickt wird.

Wenn `trans-sid` aktiviert ist, registriert **session_start()** die interne Ausgaberoutine für das Umschreiben von URLs. Verwendet ein Benutzer `ob_gzhandler` oder ähnliches mit ob_start(), ist die Reihenfolge der Augaberoutine wichtig für eine korrekte Ausgabe. Zum Beispiel muss der Benutzer `ob_gzhandler` vor Beginn der Session registrieren.

> **Anmerkung:** Die Verwendung von `zlib.output_compression` ist `ob_gzhandler` vorzuziehen.

session_destroy (PHP 4 >= 4.0.0)

Löscht alle in einer Session registrierten Daten

bool **session_destroy** (void)

session_destroy() löscht alle in Verbindung mit der aktuellen Session stehenden Daten. Mit der Session zusammenhängende Globale Variablen und das Session-Cookie werden nicht gelöscht.

Diese Funktion gibt nach erfolgtem Löschen der Daten `TRUE` zurück und `FALSE`, wenn dabei ein Fehler auftritt.

Beispiel 1. Löschen einer Session

```
<?php
// Initialisierung der Session.
// Wenn Sie session_name("irgendwas") verwenden, vergessen Sie es
// jetzt nicht!
session_start();
// Löschen aller Session-Variablen.
session_unset();
// Zum Schluß, löschen der Session.
session_destroy();

?>
```

Beispiel 2. Löschen einer Session mit $_SESSION

```
<?php

// Initialisierung der Session.
// Wenn Sie session_name("irgendwas") verwenden, vergessen Sie es
// jetzt nicht!
session_start();
// Löschen aller Session-Variablen.
unset($_SESSION);
// Zum Schluß, löschen der Session.
session_destroy();
?>
```

session_name (PHP 4 >= 4.0.0)

Liefert und/oder setzt den Namen der aktuellen Session

string **session_name** ([string Name])

session_name() gibt den Namen der aktuellen Session zurück. Wenn *Name* angegeben wird, wird der Name der aktuellen Session durch dessen Wert ersetzt.

Der Session-Name referenziert die Session-ID in Cookies und URLs. Er sollte nur alphanumerische Zeichen enthalten und sollte kurz und beschreibend sein (z.B. für Benutzer mit aktivierter Cookie-Warnung). Der Session-Name wird auf den Ausgangswert zurückgesetzt, der zu Beginn in `session.name` gespeichert wurde. Deshalb muss **session_name()** bei jeder Anfrage aufgerufen werden (und bevor session_start() oder session_register() aufgerufen wurden).

Beispiel 1. session_name() Beispiele

```
<?php
// den Session-Namen auf WebsiteID setzen
$name_vorher = session_name ("WebsiteID");
echo "Der vorhergehende Session-Name war $name_vorher<p>";
?>
```

session_module_name (PHP 4 >= 4.0.0)

Liefert und/oder setzt das aktuelle Session-Modul

string **session_module_name** ([string Modul])

session_module_name() gibt den Namen des aktuellen Session-Moduls zurück. Wenn *Modul* angegeben wird, wird stattdessen dieses Modul verwendet.

session_save_path (PHP 4 >= 4.0.0)

Liefert und/oder setzt den aktuellen Speicherpfad der Session

string **session_save_path** ([string Pfad])

session_save_path() gibt den Pfad des aktuellen Verzeichnisses zurück, das zum Speichern der Session-Daten verwendet wird. Wenn *Pfad* angegeben wird, wird der Pfad, unter dem Daten gespeichert werden, entsprechend geändert.

> **Anmerkung:** Unter manchen Betriebssystemen, wollen Sie vielleicht einen Pfad auf einem Dateisystem angeben, das sehr viele kleine Dateien effizient behandelt. Unter Linux, zum Beispiel, bietet reiserfs eventuell eine bessere Performance als ext2fs.

session_id (PHP 4 >= 4.0.0)

Liefert und/oder setzt die aktuelle Session-ID

string **session_id** ([string ID])

session_id() gibt die Session-ID der aktuellen Session zurück. Wenn *ID* angegeben wird, wird die die aktuelle Session-ID ersetzt.

Die Konstante SID kann auch dazu verwendet werden, den aktuellen Namen und die Session-ID als Zeichenkette passend zum Anhängen an URLs zu erhalten.

session_register (PHP 4 >= 4.0.0)

Registriert eine oder mehrere Variablen in der aktuellen Session

bool **session_register** (mixed Name [, mixed ...])

session_register() akzeptiert eine variable Anzahl von Argumenten, die jeweils entweder eine Zeichenkette sein können, die den Namen einer Variablen trägt, oder ein Array, das aus solchen Variablennamen oder anderen Arrays besteht. Für jeden Namen registriert **session_register()** die globale Variable mit diesem Namen in der aktuellen Session.

Achtung

Diese Funktion registriert eine *globale* Variable. Wenn Sie eine Session-Variable innerhalb einer Funktion registrieren wollen, müssen Sie sicherstellen, dass Sie sie mittels **global()** global machen oder Sie verwenden die nachstehend vermerkten Session-Arrays.

Achtung

Wenn Sie `$HTTP_SESSION_VARS`/`$_SESSION` verwenden, sollten Sie **session_register()**, session_is_registered() und session_unregister() nicht verwenden.

Diese Funktion gibt `TRUE` zurück, wenn alle Variablen erfolgreich in der Session registriert wurden.

Wenn session_start() nicht vor dieser Funktion aufgerufen wurde, erfolgt ein impliziter Aufruf von session_start() ohne Parameter.

Sie können eine Session-Variable auch erzeugen, indem Sie das entsprechende Element des `$HTTP_SESSION_VARS`- oder (PHP >= 4.1.0) `$_SESSION`-Arrays setzen.

```
$barney = "A big purple dinosaur.";
session_register("barney");

$HTTP_SESSION_VARS["zim"] = "An invader from another planet.";

# Das auto-global $_SESSION-Array wurde mit PHP 4.1.0 eingeführt
$_SESSION["spongebob"] = "He’s got square pants.";
```

Anmerkung: Gegenwärtig ist es nicht möglich, Ressourcen-Variablen in einer Session zu registrieren. Zum Beispiel können Sie nicht erwarten, dass die als Session-Variable gespeicherte Verbindungs-Kennung der zu einer Datenbank aufgebauten Verbindung bei der nächsten Wiederherstellung der Session noch gültig ist. PHP-Funktionen, die eine Ressource zurückgeben, können daran erkannt werden, dass sie einen Rückgabewert `resource` in ihrer Funktionsdefinition haben. Eine Liste der Funktionen, die Ressourcen zurückgeben befindet sich im Anhang Liste von PHP Ressourcen.

Bei Verwendung von `$_SESSION` (oder `$HTTP_SESSION_VARS` bei PHP 4.0.6 oder niedriger) weisen Sie `$_SESSION` eine Variable zu, also $_SESSION['var'] = 'ABC';

Siehe auch: session_is_registered() und session_unregister().

session_unregister (PHP 4 >= 4.0.0)

Hebt die Registrierung einer Variablen in der aktuellen Session auf

bool **session_unregister** (string Name)

session_unregister() hebt die Registrierung der globalen Variablen namens *Name* in der aktuellen Session auf.

Diese Funktion gibt `TRUE` zurück, wenn die Aufhebung der Registrierung der Variablen in der Session erfolgt ist.

Anmerkung: Benutzen Sie bei Verwendung von `$_SESSION` (oder `$HTTP_SESSION_VARS` bei PHP 4.0.6 oder niedriger) unset() um die Registrierung einer Session-Variablen aufzuheben.

Achtung

Diese Funktion löscht die *Name* entsprechende globale Variable nicht, sondern verhindert nur, dass die Variable als Teil der Session gespeichert wird. Sie müssen unset() aufrufen, um die entsprechende globale Variable zu entfernen.

Achtung

Wenn Sie `$HTTP_SESSION_VARS`/`$_SESSION` verwenden, sollten Sie session_register(), session_is_registered() und **session_unregister()** nicht verwenden.

session_unset (PHP 4 >= 4.0.0)

Befreit alle Session-Variablen

void **session_unset** (void)

Die Funktion **session_unset()** befreit alle Session-Variablen, die gegenwärtig registriert sind.

Anmerkung: Benutzen Sie bei Verwendung von `$_SESSION` (oder `$HTTP_SESSION_VARS` bei PHP 4.0.6 oder niedriger) unset() um die Registrierung einer Session-Variablen aufzuheben, also unset($_SESSION);

session_is_registered (PHP 4 >= 4.0.0)

Überprüft, ob eine Variable in einer Session registriert ist

bool **session_is_registered** (string Name)

session_is_registered() gibt `TRUE` zurück, wenn in der aktuellen Session eine Variable mit dem Namen *Name* registriert ist.

Anmerkung: Benutzen Sie bei Verwendung von `$_SESSION` (oder `$HTTP_SESSION_VARS` bei PHP 4.0.6 oder niedriger) isset() um zu prüfen, ob eine Variable in `$_SESSION` registriert ist.

Achtung

Wenn Sie `$HTTP_SESSION_VARS`/`$_SESSION` verwenden, sollten Sie session_register(), **session_is_registered()** und session_unregister() nicht verwenden.

session_get_cookie_params (PHP 4 >= 4.0.0)

Liefert die Session-Cookie Parameter

array **session_get_cookie_params** (void)

Die Funktion **session_get_cookie_params()** gibt ein Array mit der Information des aktuellen Session-Cookies zurück. Das Array enthält folgende Elemente:

- "lifetime" - Die Lebensdauer des Cookies.
- "path" - Der Pfad, unter dem Informationen gespeichert sind.
- "domain" - Die Domain des Cookies.
- "secure" - Das Cookie soll nur über sichere Verbindungen gesendet werden. (Dieses Element wurde in PHP 4.0.4 hinzugefügt.)

session_set_cookie_params (PHP 4 >= 4.0.0)

Setzt die Session-Cookie Parameter

void **session_set_cookie_params** (int Lebensdauer [, string Pfad [, string Domain]])

Setzen der Cookie-Parameter, die in der php.ini definiert sind. Diese Funktion wirkt sich nur für die Dauer des Scripts aus.

session_decode (PHP 4 >= 4.0.0)

Dekodiert die Daten einer Session aus einer Zeichenkette

bool **session_decode** (string Daten)

session_decode() dekodiert die Daten der Session in *`Daten`* und setzt die Variablen, die in der Session gespeichert sind.

session_encode (PHP 4 >= 4.0.0)

Kodiert die Daten der aktuellen Session als Zeichenkette

string **session_encode** (void)

session_encode() gibt eine Zeichenkette mit dem kodierten Inhalt der aktuellen Session zurück.

session_set_save_handler (PHP 4 >= 4.0.0)

Setzt benutzerdefinierte Session-Speicherfunktionen

void **session_set_save_handler** (string öffnen, string schließen, string lesen, string schreiben, string löschen, string gc)

session_set_save_handler() setzt die benutzerdefinierten Session-Speicherfunktionen, die zur Speicherung und Wiederherstellung von zur Session gehörigen Daten verwendet werden. Dies ist äußerst nützlich, wenn eine andere als die mit PHP-Sessions zur Verfügung stehende Art der Speicherung, z.B. die Speicherung der Session-Daten in einer lokalen Datenbank, bevorzugt wird.

Anmerkung: Sie müssen in Ihrer php.ini-Datei die Konfigurationsoption `session.save_handler` auf `user` setzen, damit **session_set_save_handler()** wirksam wird.

Anmerkung: Die "schreiben"-Routine wird nicht ausgeführt, bevor die Ausgabe abgeschlossen ist. Deshalb werden auch niemals Fehlermeldungen der "schreiben"-Routine im Browser erscheinen. Wenn die Ausgabe von Fehlermeldungen nötig ist, sollte diese stattdessen in eine Datei geschrieben werden.

Das folgende Beispiel liefert eine dateibasierte Session-Speicherung, die der Standard-Speicherroutine `files` der PHP-Sessions ähnlich ist. Dieses Beispiel kann problemlos erweitert werden, dass es Datenbank-Speicherung mittels ihrer bevorzugten PHP-gestützten Datenbank beinhaltet.

Damit die Speicherroutine funktioniert wie erwartet, muss die read-Function immer einen String-Wert zurückgeben. Wenn es keine Daten zu lesen gibt, geben Sie eine leere Zeichenkette zurück. Rückgabewerte anderer Routinen werden in boolesche Ausdrücke umgewandelt. TRUE für einen Erfolg, FALSE für einen Fehler.

Beispiel 1. session_set_save_handler() Beispiel

```
<?php
function open ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;

  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}
```

```
function close() {
  return(true);
}

function read ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return(""); // Must return "" here.
  }

}

function write ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }

}

function destroy ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/***********************************************
 * WARNUNG - Sie müssen hier irgendeine Art von *
 * Speicherbereinigungsroutine realisieren.     *
 ***********************************************/
function gc ($maxlifetime) {
  return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy",
"gc");

session_start();

// verfahren Sie mit Sessions wie gewohnt
?>
```

session_cache_limiter (PHP 4 >= 4.0.3)

Liefert und/oder setzt die aktuelle Cacheverwaltung

string **session_cache_limiter** ([string Cacheverwaltung])

session_cache_limiter() gibt die Bezeichnung der aktuellen Cacheverwaltung zurück. Wenn *Cacheverwaltung* angegeben wurde, wird diese dem neuen Wert entsprechend geändert.

Die Cacheverwaltung steuert die Cache-Kontrolle der HTTP-Header, die zum Client geschickt werden. Diese Header bestimmen die Regel, nach der der Seiteninhalt in den Cachespeicher aufgenommen wird. Wird die Cacheverwaltung beispielsweise auf `nocache` gesetzt, ist eine clientseitige Speicherung im Cache nicht erlaubt. Andererseits erlaubt der Wert `public` die Aufnahme in den Cachespeicher. Sie kann auch auf `private` gesetzt werden, was etwas restriktiver ist, als `public`.

Im `private`-Modus könnte ein zum Client gesendeter Expire-Header manche Browser, inklusive Mozilla, verwirren. Sie können dieses Problem mit dem `private_no_expire`-Modus vermeiden. In diesem Modus wird nie ein Expire-Header zum Client gesendet.

Anmerkung: `private_no_expire` wurde in PHP 4.2.0dev hinzugefügt.

Zu Beginn einer Anfrage wird die Cacheverwaltung auf den voreingestellten Wert zurückesetzt, der in `session.cache_limiter` gespeichert ist. Daher müssen Sie **session_cache_limiter()** bei jeder Anfrage aufrufen (und zwar bevor session_start() aufgerufen wird).

Beispiel 1. session_cache_limiter() Beispiel

```
<?php
# setzen der Cacheverwaltung auf 'private'
session_cache_limiter('private');
$cacheverwaltung = session_cache_limiter();
echo "Die Cacheverwaltung ist jetzt auf $cacheverwaltung gesetzt<p>";
?>
```

session_cache_expire (PHP 4 CVS only)

Liefert die aktuelle Cache-Verfallszeit

int **session_cache_expire** ([int neue_cache_verfallszeit])

session_cache_expire() gibt die aktuelle Cache-Verfallszeit zurück. Wenn neue_cache_verfallszeit gegeben ist, wird die aktuelle cache_expire durch neue_cache_verfallszeit ersetzt.

session_write_close (PHP 4 >= 4.0.4)

Speichert die Session-Daten und beendet die Session

void **session_write_close** (void)

Beendet die aktuelle Session und speichert die Session-Daten.

Session-Daten werden normalerweise nach Beenden eines Scripts gespeichert, ohne dass **session_write_close()** aufgerufen werden muss, aber da Session-Daten gesperrt werden, um gleichzeitiges Schreiben zu verhindern, kann jeweils immer nur ein Script auf eine Session einwirken. Bei der Verwendung von Framesets zusammen mit Sessions werden Sie merken, dass wegen dieser Sperrung ein Frame nach dem anderen geladen wird. Sie können die Zeit zum Laden aller Frames reduzieren, indem Sie die Session beenden, sobald alle Änderungen an den Session-Variablen durchgeführt sind.

XCI. Shared Memory Funktionen (Gemeinsamer Speicher)

Shmop ist eine Sammlung einfach anzuwendender PHP-Funktionen, mit denen man mit PHP UNIX shared-memory-Segmente beschreiben, auslesen und löschen kann. Die Funktionen können mit Windows nicht eingesetzt werden, weil Windows keinen gemeinsamen Speicher unterstützt. Um shmop zu nutzen, müssen Sie PHP mit der Option --enable-shmop in der Konfiguratinszeile kompilieren.

> **Anmerkung:** Die Funktionen, die in diesem Kapitel erklärt werden, beginnen in PHP 4.0.3 alle mit **shm_()** , aber ab Version PHP 4.0.4 und grösser, wurden diese Namen geändert und beginnen nun mit **shmop_()**.

Beispiel 1. Shared Memory Operationen - Übersicht

```
<?php

// Erstelle einen 100 Byte grossen gemeinsam genutzten Speicherblock
// mit mit der System_ID if 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
        echo "Konnte kein gemeinsames Speichersegment erstellen\n";
}

// Hole die Grösse des gemeinsamen Speicherblocks
$shm_size = shmop_size($shm_id);
echo "SHM Block mit: ".$shm_size. " Bytes wurde erstellt.\n";

// Teststring in den gemeinsamen Speicher schreiben
$shm_bytes_written = shmop_write($shm_id, "mein gemeinsamer Speicher", 0);
if($shm_bytes_written != strlen("mein gemeinsamer Speicher")) {
        echo "Konnte nicht ermitteln, wieviele Bytes geschrieben wurden\n";
}

// Den Teststring wieder auslesen
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
        echo "Konnte nicht aus dem gemeinsamen Speicher lesen\n";
}
echo "Die Daten im gemeinsamen Speicher waren: ".$my_string."\n";

// Den Speicherblock löschen und den gemeinsamen Speicher schliessen
if(!shmop_delete($shm_id)) {
        echo "Konnte den gemeinsamen Speicherblock nicht zum Löschen markieren.";
}
shmop_close($shm_id);

?>
```

shmop_open (PHP 4 >= 4.0.4)

Erstellt oder öffnet einen gemeinsamen Speicherblock

int **shmop_open** (int key, string flags, int mode, int size)

shmop_open() Kann einen gemeinsamen Speicherbereich erstellen oder öffnen

shmop_open() benötigt 4 Parameter: key, die System-ID des gemeinsam genutzten Speicherblocks, dieser Parameter kann als Dezimal- oder Hexadezimalzahl übergeben werden. Der zweite Parameter bezeichnet die Flags, die man benutzen kann:

- "a" zum Zugriff (access) auf einen gemeinsamen Speicherblock (setzt IPC_EXCL) Benutzen Sie dieses Flag, wenn Sie einen bestehenden gemeinsamen Speicherblock öffnen wollen.
- "c" zum Erstellen (create) eines gemeinsamen Speicherblocks (setzt IPC_CREATE) Benutzen Sie dieses Flag, wenn Sie einen gemeinsamen Speicherblock erstellen wollen.

Der dritte Parameter ist der Modus, der die Zugriffsrechte, die Sie dem gemeinsamen Speicher zuweisen wollen, angibt. Das sind dieselben Zugriffsrechte wie für Dateien. Zugriffsrechte müssen oktal angegeben werden, z.B. 0644. Der letzte Parameter gibt die Grösse des gemeinsamen Speichers, den Sie erstellen wollen, in Byte, an.

Anmerkung: Beachten Sie: Der dritte und vierte Parameter muss als 0 angegeben werden, wenn Sie ein existierendes gemeinsames Speichersegment öffnen wollen. Bei Erfolg liefert die Funktion **shmop_open()** eine ID zurück, die Sie zum Zugriff auf den erstellten gemeinsamen Speicher benutzen können.

Beispiel 1. Einen neuen gemeinsamen Speicherblock erstellen

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

In diesem Beispiel wurde ein gemeinsamer Speicherblock mit der System-ID 0x0fff erstellt.

shmop_read (PHP 4 >= 4.0.4)

Daten aus dem gemeinsamen Speicherblock auslesen

string **shmop_read** (int shmid, int start, int count)

shmop_read() liest einen String aus dem gemeinsamen Speicher.

shmop_read() braucht 3 Parameter: shmid, die Kennung des gemeinsamen Speicherblocks, der mit **shmop_open()** erzeugt wurde, start gibt den Offset an, von wo ab gelesen werden soll und count gibt die Anzahl der zu lesenden Bytes an.

Beispiel 1. Aus dem gemeinsamen Speicher lesen

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

In diesem Beispiel werden 50 Bytes aus dem gemeinsamen Speicherblock in die Variable `$shm_data` eingelesen.

shmop_write (PHP 4 >= 4.0.4)

Schreibt Daten in einen gemeinsamen Speicherblock

int **shmop_write** (int shmid, string data, int offset)

shmop_write() schreibt Daten in einen gemeinsamen Speicherbereich.

shmop_write() braucht 3 Parameter:shmid, die Kennung des gemeinsamen Speicherblocks, der mit **shmop_open()** erzeugt wurde, data, ein String, der in den gemeinsamen Speicher geschrieben werden soll und offset, der angibt, von wo ab in den gemeinsamen Speicherblock geschrieben werden soll.

Beispiel 1. In den gemeinsamen Speicher schreiben

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

In diesem Beispiel werden die Daten der Variablen `$my_string` in den gemeinsamen Speicherblock geschrieben. Die Variable `$shm_bytes_written` enthält die Anzahl der geschriebenen Bytes.

shmop_size (PHP 4 >= 4.0.4)

Liefert die Grösse des gemeinsamen Speicherblocks

int **shmop_size** (int shmid)

shmop_size() wird benutzt, um die Grösse eines gemeinsamen Speicherblocks in Byte zu ermitteln.

shmop_size() benötigt shmid, die Kennung des gemeinsamen Speicherblocks, der mit **shmop_open()** erzeugt wurde. Die Funktion liefert einen Integerwert zurück, welcher angibt, wieviele Bytes von dem gemeinsamen Speicher belegt werden.

Beispiel 1. Die Grösse des gemeinsamen Speicherblocks ermitteln

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

In dem Beispiel erhält die Variable `$shm_size` die Grösse des durch `$shm_id` bezeichneten gemeinsamen Speicherblocks in Byte.

shmop_delete (PHP 4 >= 4.0.4)

Einen gemeinsamen Speicherblock löschen

int **shmop_delete** (int shmid)

shmop_delete() wird benutzt, um einen gemeinsamen Speicherbereich zu löschen.

shmop_delete() benötigt shmid, die Kennung des gemeinsamen Speicherblocks, der mit **shmop_open()** erzeugt wurde. Bei Erfolg wird 1 zurückgegeben, bei Scheitern 0.

Beispiel 1. einen gemeinsamen Speicherblock löschen

```
<?php
shmop_delete($shm_id);
?>
```

In diesem Beispiel wird der gemeinsame Speicherblock mit dem Bezeichner `$shm_id` gelöscht.

shmop_close (PHP 4 >= 4.0.4)

Schliesst einen gemeinsamen Speicherblock

int **shmop_close** (int shmid)

shmop_close() wird benutzt, um einen gemeinsamen Speicherblock zu schliessen.

shmop_close() benötigt shmid, die Kennung des gemeinsamen Speicherblocks, der mit **shmop_open()** erzeugt wurde.

Beispiel 1. Einen gemeinsamen Speicherblock schliessen

```
<?php
shmop_close($shm_id);
?>
```

In diesem Beispiel wird der gemeinsame Speicherblock mit dem Bezeichner `$shm_id` geschlossen.

XCII. Shockwave Flash Funktionen

PHP ermöglicht es Shockwave Flash Dateien mit Paul Haeberli's libswf Modul zu erzeugen. Sie können das libswf-Modul unter http://reality.sgi.com/grafica/flash/ herunterladen. Wenn Sie das Modul geladen haben, brauchen Sie nur noch PHP mit dem Modul zu konfigurieren (`--with-swf[=DIR]`), wobei DIR für das Verzeichnis steht, welches die Verzeichnisse include und lib beinhaltet. Das include-Verzeichnis muss die swf.h-Datei und das lib-Verzeichnis die libswf.a-Datei enthalten. Wenn Sie die Distribution entpacken, werden beide Dateien in ein Verzeichnis entpackt. Daher müssen Sie die Dateien manuell in die richtigen Verzeichnisse kopieren.

Haben Sie PHP erfolgreich mit Shockwave Flash Unterstützung installiert, können Sie Shockwave Flash-Dateien via PHP erstellen. Sie werden überrascht sein von den Möglichkeiten, die sich Ihnen bieten. Schauen Sie sich den folgenden Quellcode-Auschnitt an:

Beispiel 1. SWF Beispiel

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}
```

```
swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

Dieses Beispiel wird die Animation generieren, die Sie unter folgender Adresse (URL) ansehen können: url (http://www.designmultimedia.com/swfphp/test.swf).

Anmerkung: SWF-Unterstützung wurde in PHP4 RC2 hinzugefügt.

swf_openfile (PHP 4 >= 4.0.0)

Öffnet eine neue Shockwave Flash Datei

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b)

Die Funktion **swf_openfile()** öffnet eine neue SWF-Datei mit dem Namen *filename*, einer Breite von *width*,einer Höhe von *height*, einer Frame-Rate von *framerate* und einem Hintergrund mit einem roten Farbwert von *r*, einem grünen Farbwert von *g* und einem blauen Farbwert von *b*.

Die Funktion **swf_openfile()** muss die erste Funtkion sein, sie Sie aufrufen, andernfalls wird Ihr Skript einen Segfault verursachen. Wenn Sie den Output des Skriptes auf dem Bildschirm ausgeben wollen, benutzen Sie filename: "php://stdout" (Unterstützung ab PHP 4.0.1 und höher).

swf_closefile (PHP 4 >= 4.0.0)

Schließt die aktuelle SWF-Datei

void **swf_closefile** (void)

Schließt eine Datei die zuvor mit **swf_openfile ()** geöffnet wurde.

swf_labelframe (PHP 4 >= 4.0.0)

Beschriftet den aktuellen Frame

void **swf_labelframe** (string name)

Beschriftet den aktuellen Frame mit der Bezeichnung, die durch das Parameter *name* angegeben wurde.

swf_showframe (PHP 4 >= 4.0.0)

Zeigt den aktuellen Frame an

void **swf_showframe** (void)

Die swf_showframe() Funktion gibt den aktuellen Frame aus.

swf_setframe (PHP 4 >= 4.0.0)

Wechselt zu einem spezifizierten Frame

void **swf_setframe** (int framenumber)

Die **swf_setframe()** Funktion ändert den aktuellen Frame in den mit *framenumber* spezifizierten Frame.

swf_getframe (PHP 4 >= 4.0.0)

Holt die Frame-Nummer des aktuellen Frames

int **swf_getframe** (void)

Die **swf_getframe()** Funtkion holt die Nummer des aktuellen Frames.

swf_mulcolor (PHP 4 >= 4.0.0)

Setzt die "global multiply"-Farbe auf den spezifizierten rgba-Wert

void **swf_mulcolor** (float r, float g, float b, float a)

Die **swf_mulcolor()** Funktion setzt die "global multiply"-Farbe auf die mit *rgba* spezifizierte Farbe. Diese Farbe wird (implizit) von den Funktionen swf_placeobject(), swf_modifyobject() und **swf_addbuttonrecord()** verwendet. Die Farbe des Objekts wird vervielfacht mit dem Wert von *rgba* , wenn das Objekt auf dem bildschirm ausgegeben wird.

Anmerkung: Das *rgba* Parameter kann sowohl negative wie auch positive Werte annehmen.

swf_addcolor (PHP 4 >= 4.0.0)

Setzt die "global add"-Farbe auf den spezifizierten rgba-Wert

void **swf_addcolor** (float r, float g, float b, float a)

Die Funktion **swf_addcolor()** setzt die "global add"-Farbe auf den mit *rgba* spezifizierten Farbwert. Diese Farbe wird (implizit) von den Funktionen **swf_placeobject()**, swf_modifyobject() und swf_addbuttonrecord() verwendet. Die Farbe wird dem Objekt durch den *rgba*-Wert hinzugefügt, wenn es auf dem Bildschirm ausgegeben wird.

Anmerkung: Das *rgba* Parameter kann sowohl negative wie auch positive Werte annehmen.

swf_placeobject (PHP 4 >= 4.0.0)

Place an object onto the screen

void **swf_placeobject** (int objid, int depth)

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by swf_mulcolor()) and the current addcolor (specified by swf_addcolor()) to color the object and it uses the current matrix to position the object.

Anmerkung: Full RGBA colors are supported.

swf_modifyobject (PHP 4 >= 4.0.0)

Modify an object

void **swf_modifyobject** (int depth, int how)

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function swf_mulcolor()) and addcolor (specified by the function swf_addcolor()) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_removeobject (PHP 4 >= 4.0.0)

Remove an object

void **swf_removeobject** (int depth)

Removes the object at the depth specified by *depth*.

swf_nextid (PHP 4 >= 4.0.0)

Returns the next free object id

int **swf_nextid** (void)

The **swf_nextid()** function returns the next available object id.

swf_startdoaction (PHP 4 >= 4.0.0)

Start a description of an action list for the current frame

void **swf_startdoaction** (void)

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_actiongotoframe (PHP 4 >= 4.0.0)

Play a frame and then stop

void **swf_actiongotoframe** (int framenumber)

The **swf_actionGotoFrame()** function will go to the frame specified by `framenumber`, play it, and then stop.

swf_actiongeturl (PHP 4 >= 4.0.0)

Get a URL from a Shockwave Flash movie

void **swf_actiongeturl** (string url, string target)

The **swf_actionGetUrl()** function gets the URL specified by the parameter `url` with the target `target`.

swf_actionnextframe (PHP 4 >= 4.0.0)

Go foward one frame

void **swf_actionnextframe** (void)

Go foward one frame.

swf_actionprevframe (PHP 4 >= 4.0.0)

Go backwards one frame

void **swf_actionprevframe** (void)

swf_actionplay (PHP 4 >= 4.0.0)

Start playing the flash movie from the current frame

void **swf_actionplay** (void)

Start playing the flash movie from the current frame.

swf_actionstop (PHP 4 >= 4.0.0)

Stop playing the flash movie at the current frame

void **swf_actionstop** (void)

Stop playing the flash movie at the current frame.

swf_actiontogglequality (PHP 4 >= 4.0.0)

Toggle between low and high quality

void **swf_actiontogglequality** (void)

Toggle the flash movie between high and low quality.

swf_actionwaitforframe (PHP 4 >= 4.0.0)

Skip actions if a frame has not been loaded

void **swf_actionwaitforframe** (int framenumber, int skipcount)

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the `framenumber` parameter has been loaded, if not it will skip the number of actions specified by the `skipcount` parameter. This can be useful for "Loading..." type animations.

swf_actionsettarget (PHP 4 >= 4.0.0)

Set the context for actions

void **swf_actionsettarget** (string target)

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actiongotolabel (PHP 4 >= 4.0.0)

Display a frame with the specified label

void **swf_actiongotolabel** (string label)

The **swf_actionGotoLabel()** function displays the frame with the label given by the `label` parameter and then stops.

swf_enddoaction (PHP 4 >= 4.0.0)

End the current action

void **swf_enddoaction** (void)

Ends the current action started by the swf_startdoaction() function.

swf_defineline (PHP 4 >= 4.0.0)

Define a line

void **swf_defineline** (int objid, float x1, float y1, float x2, float y2, float width)

The **swf_defineline()** defines a line starting from the x coordinate given by `x1` and the y coordinate given by `y1` parameter. Up to the x coordinate given by the `x2` parameter and the y coordinate given by the `y2` parameter. It will have a width defined by the `width` parameter.

swf_definerect (PHP 4 >= 4.0.0)

Define a rectangle

void **swf_definerect** (int objid, float x1, float y1, float x2, float y2, float width)

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, `x1`, and the y, `y1`. And a lower right hand coordinate given by the x coordinate, `x2`, and the y coordinate, `y2` . Width of the rectangles border is given by the `width` parameter, if the width is 0.0 then the rectangle is filled.

swf_definepoly (PHP 4 >= 4.0.0)

Define a polygon

void **swf_definepoly** (int objid, array coords, int npoints, float width)

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter `coords`). The parameter `npoints` is the number of overall points that are contained in the array given by `coords`. The `width` is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_startshape (PHP 4 >= 4.0.0)

Start a complex shape

void **swf_startshape** (int objid)

The **swf_startshape()** function starts a complex shape, with an object id given by the `objid` parameter.

swf_shapelinesolid (PHP 4 >= 4.0.0)

Set the current line style

void **swf_shapelinesolid** (float r, float g, float b, float a, float width)

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapefilloff (PHP 4 >= 4.0.0)

Turns off filling

void **swf_shapefilloff** (void)

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid (PHP 4 >= 4.0.0)

Set the current fill style to the specified color

void **swf_shapefillsolid** (float r, float g, float b, float a)

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapefillbitmapclip (PHP 4 >= 4.0.0)

Set current fill mode to clipped bitmap

void **swf_shapefillbitmapclip** (int bitmapid)

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile (PHP 4 >= 4.0.0)

Set current fill mode to tiled bitmap

void **swf_shapefillbitmaptile** (int bitmapid)

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapemoveto (PHP 4 >= 4.0.0)

Move the current position

void **swf_shapemoveto** (float x, float y)

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_shapelineto (PHP 4 >= 4.0.0)

Draw a line

void **swf_shapelineto** (float x, float y)

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapecurveto (PHP 4 >= 4.0.0)

Draw a quadratic bezier curve between two points

void **swf_shapecurveto** (float x1, float y1, float x2, float y2)

The **swf_shapecurveto()** function draws a quadratic bezier curve from the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapecurveto3 (PHP 4 >= 4.0.0)

Draw a cubic bezier curve

void **swf_shapecurveto3** (float x1, float y1, float x2, float y2, float x3, float y3)

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapearc (PHP 4 >= 4.0.0)

Draw a circular arc

void **swf_shapearc** (float x, float y, float r, float ang1, float ang2)

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_endshape (PHP 4 >= 4.0.0)

Completes the definition of the current shape

void **swf_endshape** (void)

The **swf_endshape()** completes the definition of the current shape.

swf_definefont (PHP 4 >= 4.0.0)

Defines a font

void **swf_definefont** (int fontid, string fontname)

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_setfont (PHP 4 >= 4.0.0)

Change the current font

void **swf_setfont** (int fontid)

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_fontsize (PHP 4 >= 4.0.0)

Change the font size

void **swf_fontsize** (float size)

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant (PHP 4 >= 4.0.0)

Set the font slant

void **swf_fontslant** (float slant)

Set the current font slant to the angle indicated by the `slant` parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking (PHP 4 >= 4.0.0)

Set the current font tracking

void **swf_fonttracking** (float tracking)

Set the font tracking to the value specified by the `tracking` parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getfontinfo (PHP 4 >= 4.0.0)

The height in pixels of a capital A and a lowercase x

array **swf_getfontinfo** (void)

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_definetext (PHP 4 >= 4.0.0)

Define a text string

void **swf_definetext** (int objid, string str, int docenter)

Define a text string (the `str` parameter) using the current font and font size. The `docenter` is where the word is centered, if `docenter` is 1, then the word is centered in x.

swf_textwidth (PHP 4 >= 4.0.0)

Get the width of a string

float **swf_textwidth** (string str)

The **swf_textwidth()** function gives the width of the string, `str`, in pixels, using the current font and font size.

swf_definebitmap (PHP 4 >= 4.0.0)

Define a bitmap

void **swf_definebitmap** (int objid, string image_name)

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_getbitmapinfo (PHP 4 >= 4.0.0)

Get information about a bitmap

array **swf_getbitmapinfo** (int bitmapid)

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_startsymbol (PHP 4 >= 4.0.0)

Define a symbol

void **swf_startsymbol** (int objid)

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_endsymbol (PHP 4 >= 4.0.0)

End the definition of a symbol

void **swf_endsymbol** (void)

The **swf_endsymbol()** function ends the definition of a symbol that was started by the swf_startsymbol() function.

swf_startbutton (PHP 4 >= 4.0.0)

Start the definition of a button

void **swf_startbutton** (int objid, int type)

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_addbuttonrecord (PHP 4 >= 4.0.0)

Controls location, appearance and active area of the current button

void **swf_addbuttonrecord** (int states, int shapeid, int depth)

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Beispiel 1. swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_oncondition (PHP 4 >= 4.0.0)

Describe a transition used to trigger an action list

void **swf_oncondition** (int transition)

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)

- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_endbutton (PHP 4 >= 4.0.0)

End the definition of the current button

void **swf_endbutton** (void)

The **swf_endButton()** function ends the definition of the current button.

swf_viewport (PHP 4 >= 4.0.0)

Select an area for future drawing

void **swf_viewport** (double xmin, double xmax, double ymin, double ymax)

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

swf_ortho (PHP 4)

Defines an orthographic mapping of user coordinates onto the current viewport

void **swf_ortho** (double xmin, double xmax, double ymin, double ymax, double zmin, double zmax)

The **swf_ortho()** funcion defines a orthographic mapping of user coordinates onto the current viewport.

swf_ortho2 (PHP 4 >= 4.0.0)

Defines 2D orthographic mapping of user coordinates onto the current viewport

void **swf_ortho2** (double xmin, double xmax, double ymin, double ymax)

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_perspective (PHP 4 >= 4.0.0)

Define a perspective projection transformation

void **swf_perspective** (double fovy, double aspect, double near, double far)

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

Anmerkung: Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_polarview (PHP 4 >= 4.0.0)

Define the viewer's position with polar coordinates

void **swf_polarview** (double dist, double azimuth, double incidence, double twist)

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_lookat (PHP 4 >= 4.0.0)

Define a viewing transformation

void **swf_lookat** (double view_x, double view_y, double view_z, double reference_x, double reference_y, double reference_z, double twist)

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_pushmatrix (PHP 4 >= 4.0.0)

Push the current transformation matrix back unto the stack

void **swf_pushmatrix** (void)

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_popmatrix (PHP 4 >= 4.0.0)

Restore a previous transformation matrix

void **swf_popmatrix** (void)

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_scale (PHP 4 >= 4.0.0)

Scale the current transformation

void **swf_scale** (double x, double y, double z)

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_translate (PHP 4 >= 4.0.0)

Translate the current transformations

void **swf_translate** (double x, double y, double z)

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_rotate (PHP 4 >= 4.0.0)

Rotate the current transformation

void **swf_rotate** (double angle, string axis)

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_posround (PHP 4 >= 4.0.0)

Enables or Disables the rounding of the translation when objects are placed or moved

void **swf_posround** (int round)

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

XCIII. SNMP Funktionen

Um die SNMP Funktionen unter Unix nutzen zu können muss das UCD SNMP (http://net-snmp.sourceforge.net/) Paket installiert sein. Unter Windows sind diese Funktionen nur unter NT verfügbar, nicht aber unter Win95/98.

WICHTIG: Um das UCD SNMP Paket zu nutzen muss NO_ZEROLENGTH_COMMUNITY als 1 definiert werden bevor man es kompiliert. Nache dem Konfigureieren UCD SNMP muss in der Datei config.h die Zeile mit NO_ZEROLENGTH_COMMUNITY auskommentiert werden. Die Zeile sollte danach so aussehen:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Wenn seltsame "segmentation faults" in Verbindung mit SNMP Kommandos auftauchen, wurden die obigen Anweisungen nicht korrekt befolgt. Als eine Alternative zum Neukompilieren des Packets, kann mann auch PHP mit der option --enable-ucd-snmp-hack kompilieren.

snmpget (PHP 3, PHP 4 >= 4.0.0)

Ein SNMP Objekt holen

string **snmpget** (string hostname, string community, string object_id [, int timeout [, int retries]])

Gibt den Wert eines SNMP Objektes zurück, bei Erfolg und `FALSE` bei einem Fehler.

Die **snmpget()** Funktion wird verwendet um den Wert eines SNMP Objekts zu ermitteln, das mit *`object_id`* näher bestimmt ist. Der SNMP "agent" wird mit *`hostname`* deklariert und die "read community" wird in *`community`* angegeben.

```
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
```

snmpset (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Setzt ein SNMP Objekt

bool **snmpset** (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]])

Setzt den angegebenen SNMP Objekt Wert, gibt `TRUE` bei Erfolg und `FALSE` bei einem Fehler zurück. Sets the specified SNMP object value, returning `TRUE` on success

Die Funktion **snmpset()** wird benutzt, um den Wert eines SNMP Objekts zu setzen, der mit *`object_id`* angegeben wird. Der SNMP Agent wird bei *`hostname`* angegeben und die Lese-"Community" mit dem *`community`* Parameter.

snmpwalk (PHP 3, PHP 4 >= 4.0.0)

Holt alle SNMP Objekte eines Agenten

array **snmpwalk** (string hostname, string community, string object_id [, int timeout [, int retries]])

Gibt einen Array von SNMP Objekten zurück, der mit der **object_id()** als Wurzel beginn, oder `FALSE` bei einem Fehler.

Die **snmpwalk()** Funktion wird verwendet um alle Werte von dem SNMP Agenten zu lesen, der bei *`hostname`* angegebn wurde. Der *`Community`* Parameter bestimm die Lese-"Community" für diesen Agenten. Ein `NULL` *`object_id`* wird als Wurzel des SNMP Objektbaums verwendet und alle Objekts unter diesem Baum als ein Array zurückgegeben. Wenn *`object_id`* angegeben wird, werden alle SNMP Objekte unterhalb der *`object_id`* zurückgegeben.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Der obige Funktionsaufruf gibt alle SNMP Objekte des Agenten, der auf localhost läuft, zurück. Man kann mit einer Schleife alle Werte durchlaufen.

```
for ($i=0; $i < count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Abfrage über einen Baum einer Netzwerkeinheit.

array **snmpwalkoid** (string hostname, string community, string object_id [, int timeout [, int retries]])

Gibt ein assoziatives Array mit den Objekt Ids und den dazugehörigen Objektwerten zurück, der mit der *object_id* als Wurzel beginnt, oder `FALSE` bei einem Fehler.

Die **snmpwalkoid()** Funktion wird verwendet um alle Werte von dem SNMP Agenten zu lesen, der bei *hostname* angegebn wurde. Der *Community* Parameter bestimm die Lese-"Community" für diesen Agenten. Ein `NULL` *object_id* wird als Wurzel des SNMP Objektbaums verwendet und alle Objekts unter diesem Baum als ein Array zurückgegeben. Wenn *object_id* angegeben wird, werden alle SNMP Objekte unterhalb der *object_id* zurückgegeben.

Die Existenz von **snmpwalkoid()** und snmpwalk() hat historische Gründe. Beide Funktionen gibt es aus Kompatiblitätsgründen.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

Der obige Funktionsaufruf gibt alle SNMP Objekte des Agenten, der auf localhost läuft, zurück. Man kann mit einer Schleife alle Werte durchlaufen.

```
for ($i=0; $i < count($a); $i++) {
    echo $a[$i];
}
```

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Holt den aktuellen Wert der quick_print Einstellung der UCD Bibliothek

boolean **snmp_get_quick_print** (void)

Gibt den aktuellen Wert von quick_print der UCD Bibliothek zurück. quick_print ist standartmäßig abgeschaltet.

```
$quickprint = snmp_get_quick_print();
```

Der obige Funktionsaufruf gibt `FALSE` zurück, wenn quick_print ausgeschaltet ist und `TRUE`, wenn quick_print eingeschaltet ist.

Die **snmp_get_quick_print()** Funktion ist nur bei Verwendung der SNMP Bibliothek verfügbar, jedoch nicht bei Verwendung der Windows SNMP Bibliothek.

Siehe: snmp_set_quick_print() für eine Vollständie Beschreibung was quick_print bewirkt..

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setzt den Wert von quick_print innerhalb der UCD SNMP Bibliothek.

void **snmp_set_quick_print** (boolean quick_print)

Setzt den Wert von quick_print innerhalb der UCD SNMP Bibliothek. Falls esgesetzt ist (1), wird die SNMP Bibliothek 'quick printed' Werte zurückgeben. Das Bedeutet, dass nur der Wert ausgegeben wird. Wenn quick_print nicht aktiviert ist, (Standarteinstellung) gibt die UCD SNMP Bibliothek extra Informationen aus, die den Typ des Wertes (z.B. IpAddress oder OID) enhalten. Zusätzlich gibt die Bibliothek Hex-Werte für alle Werte, die weniger als vier Zeichen haben, aus, wenn quick_print nicht aktiviert ist.

quick_print wird oft verwendet um die Informationen, die zurückgegeben werden, weiterzuverwenden anstatt sie auszugeben.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

Der erste Wert, der ausgegeben wird, ist:: 'Timeticks: (0) 0:00:00.00', waehrend mit aktiviertem quick_print, nur '0:00:00.00' ausgegeben wird.

Standartmäßig gibt die UCD SNMP umfangreiche Werte zurück, quick_print wird verwendet um nur die Werte zurückzugeben.

Momentan werden Strings noch mit extra Quotierungen zurückgegeben, dies wird in Zukunft behoben werden.

Die **snmp_set_quick_print()** Funktion ist nur bei Verwendung der SNMP Bibliothek verfügbar, jedoch nicht bei Verwendung der Windows SNMP Bibliothek.

XCIV. Socket Funktionen

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

Die Socket-Erweiterung stellt eine hardwarenahe Schnittstelle zu den Datenaustauschfunktionen einer Socket-Verbindung zur Verfügung und bietet die Möglichkeit, entweder als Socket-Server oder als Client zu agieren.

Die hier beschriebenen Socket Funktionen sind Teil einer PHP Erweiterung, die beim Kompilieren mit der Option `--enable-sockets` beim Befehl **configure** eingebunden werden müssen.

Für eine allgemeinere clientseitige Socket Schnittstelle siehe fsockopen() und pfsockopen().

Wenn man diese Funktionen benutzt, darf man nicht vergessen, dass, obwohl viele der Funktionen gleichlautende Namen wie ihre Entsprechungen in C haben, sie dennoch oft unterschiedliche Deklarationen haben. Bitte lesen Sie die Deklarationen, um Verwirrung zu vermeiden.

Das heisst, wem Socketprogrammierung fremd ist, der findet eine grosse Menge nützlicher Materialien in den entsprechenden Unix manpages und es gibt jede Menge von Tutorials und Informationen über Socketprogrammierung in C im Internet. Vieles davon kann mit leichten Änderungen für die Socketprogrammierung mit PHP benutzt werden.

Beispiel 1. Socket Beispiel: Einfacher TCP/IP Server

Dieses Beispiel zeigt einen einfachen Echo-Server. Passen Sie die Variablen `address` und `port` an Ihre Systemumgebung an und führen Sie das Skript aus. Dann können Sie sich mit einem Befehl wie: **telnet 192.168.1.53 10000** (bei dem adress und port an Ihre Umgebung angepasst ist) mit dem Server verbinden. Alles, was Sie eingeben, wird an den Server geschickt und von diesem wieder an Sie zurückgegeben. Um die Verbindung zu trennen, geben sie ’quit’ ein.

```
<?php
error_reporting (E_ALL);

/* Warten auf Verbindungen */
set_time_limit (0);

$address = ’192.168.1.53’;
$port = 10000;

if (($sock = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket() fehlgeschlagen: Grund: " . strerror ($sock) . "\n";
}

if (($ret = bind ($sock, $address, $port)) < 0) {
    echo "bind() fehlgeschlagen: Grund: " . strerror ($ret) . "\n";
}

if (($ret = listen ($sock, 5)) < 0) {
    echo "listen() fehlgeschlagen: Grund: " . strerror ($ret) . "\n";
}
```

```
do {
    if (($msgsock = accept_connect($sock)) < 0) {
        echo "accept_connect() fehlgeschlagen: Grund: " . strerror ($msgsock) . "\n";
        break;
    }
    do {
        $buf = ”;
        $ret = read ($msgsock, $buf, 2048);
        if ($ret < 0) {
            echo "read() fehlgeschlagen: Grund: " . strerror ($ret) . "\n";
            break 2;
        }
        if ($ret == 0) {
            break 2;
        }
        $buf = trim ($buf);
        if ($buf == ’quit’) {
            close ($msgsock);
            break 2;
        }
        $talkback = "PHP: Du sagtest ’$buf’.\n";
        write ($msgsock, $talkback, strlen ($talkback));
        echo "$buf\n";
    } while (true);
    close ($msgsock);
} while (true);

close ($sock);
?>
```

Beispiel 2. Socket Beispiel: Einfacher TCP/IP Client

Dieses Beispiel zeigt einen einfachen HTTP-Client. Er verbindet sich mit einer Seite, schickt einen HEAD-Request, gibt die Antwort zurueck und beendet das Skript.

```
<?php
error_reporting (E_ALL);

echo "<h2>TCP/IP Verbindung</h2>\n";

/* Die Portnummer des WWW Service holen. */
$service_port = getservbyname (’www’, ’tcp’);

/* Die IP-Adresse des Zielhosts holen. */
$address = gethostbyname (’www.php.net’);

/* Eine TCP/IP Socketverbindung aufbauen. */
$socket = socket (AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket() fehlgeschlagen: Grund: " . strerror ($socket) . "\n";
} else {
    "socket() ausgefuehrt: " . strerror ($socket) . "\n";
}
```

```
echo "Verbindungsversuch mit ’$address’ on port ’$service_port’...";
$result = connect ($socket, $address, $service_port);
if ($result < 0) {
    echo "connect() fehlgeschlagen.\nGrund: ($result) " . strerror($result) . "\n";
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = ”;

echo "HTTP HEAD request senden...";
write ($socket, $in, strlen ($in));
echo "OK.\n";

echo "Antwort lesen:\n\n";
while (read ($socket, $out, 2048)) {
    echo $out;
}

echo "Socketverbindung schliessen...";
close ($socket);
echo "OK.\n\n";
?>
```

accept_connect (4.0.2 - 4.0.6 only)

Weist den Socket an, hereinkommende Verbindungen zu bearbeiten

int **accept_connect** (int socket)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Nachdem der Socket `socket` mit der socket() erzeugt wurde, mit bind() mit einem Namen verknuepft wurde und mit listen() angewiesen wurde, auf hereinkommende Verbindungsanforderungen zu horchen, wird der Socket mit dieser Funktion angewiesen, hereinkommende Verbindungsanforderungen zuzulassen und zu bearbeiten. Wenn ein Verbindungsaufbau erfolgreich war, wird ein neuer Socket-Deskriptot zurückgegeben, welcher zur Kommunikation benutzt werden kann. Wenn es mehrere Verbindungsanforderungen an diesem Socket gibt, werden diese in eine Warteschlange gestellt und die erste Verbindungsanforderung wird bedient. Falls es keine aktuellen Verbindungsanforderungen gibt, blockiert die Funktion **accept_connect()** solange, bis eine Verbindungsanforderung kommt. Falls `socket` mit der Funktion socket_set_blocking() oder mit **set_nonblock()** auf 'nicht blockieren' gesetzt wurde, wird ein Fehlercode zurückgegeben.

Der Socket-Deskriptor, der von **accept_connect()** zurückgegeben wird, soll nicht dazu benutzt werden, um neue Verbindungsanforderungen zuzulassen. Der originäre wartende Socket `socket` bleibt geöffnet und kann wiederbenutzt werden.

Gibt einen neuen Socket-Deskriptor zurück oder einen negativen Fehlercode, falls ein Fehler auftritt. Dieser Fehlercode kann an die Funktion strerror() übergeben werden, um eine textuelle Beschreibung des Fehlers zu erhalten.

Siehe auch: bind(), connect(), listen(), socket(), socket_get_status(), und strerror().

bind (4.0.2 - 4.0.6 only)

Verknüpft einen Namen mit einem Socket

int **bind** (int socket, string address [, int port])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

bind() verknüpft den Namen, der in `address` angegeben ist mit dem Socket, der durch `socket` bestimmt wird. Dies muss ein gültiger Socket-Deskriptor sein, der mit der Funktion socket() erzeugt wurde.

Der Parameter *address* ist entweder eine IP-Adresse, angegeben in Punktnotation (z.B. `127.0.0.1`), falls der Socket zur `AF_INET` -Familie gehört; oder ein Pfadname eines Unix-domain socket, falls er zur `AF_UNIX` - Familie gehört.

Der Parameter *port* wird nur benutzt, wenn eine Verbindung zu einem `AF_INET` -Socket aufgebaut wird und bestimmt den Port des entfernten Hosts, zu dem die Verbindung hergestellt werden soll.

Gibt bei Erfolg `NULL` zurück, ansonsten einen negativen Fehlercode. Dieser Fehlercode kann an die Funktion strerror() übergeben werden, um eine textuelle Beschreibung des Fehlers zu erhalten.

Siehe auch: accept_connect(), connect(), listen(), socket(), socket_get_status(), und strerror().

close (4.0.2 - 4.0.6 only)

Schliesst einen Datei-Deskriptor

bool **close** (int socket)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

close() schliesst den Datei- (oder Socket-) Deskriptor, der durch *socket* gegeben ist.

Beachten sie, dass **close()** nicht mit PHP Dateideskriptoren, die mit fopen(), popen(), fsockopen() oder **psockopen()** erzeugt wurden, benutzt werden sollte, sondern nur für Dateideskriptoren, die mit socket() oder accept_connect() erzeugt wurden.

Gibt bei Erfolg `TRUE` zurück oder, falls ein Fehler auftrat, `FALSE`. (z.B. *socket* is invalid).

Siehe auch: bind(), listen(), socket(), socket_get_status(), und strerror().

connect (4.0.2 - 4.0.6 only)

Baut eine Verbindung zu einem Socket auf

int **connect** (int socket, string address [, int port])

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Baut mit Hilfe eines gültigen Socket-Deskriptors *socket*, der von der Funktion socket() zurückgegeben wurde, eine Verbindung zu dem Socket auf.

Der Parameter *address* ist entweder eine IP-Adresse in Punktnotation (e.g. `127.0.0.1`), falls der Socket zur `AF_INET` -Familie gehört, oder der Pfadname eines Unix-domain socket, falls der Socket zur `AF_UNIX` -Familie gehört.

Der Parameter *port* wird nur benutzt, falls eine Verbindung zu einem `AF_INET` Socket aufgebaut wird, und bestimmt den Port des entfernten Hosts, zu dem die Verbindung aufgebaut werden soll.

Gibt bei Erfolg `NULL` zurück oder einen negativen Fehlercode, falls ein Fehler aufgetreten ist.Dieser Fehlercode kann an die Funktion strerror() übergeben werden, um eine textuelle Beschreibung des Fehlers zu erhalten.

Siehe auch: bind(), listen(), socket(), socket_get_status(), und strerror().

listen (4.0.2 - 4.0.6 only)

Horcht auf eine Verbindungsanforderung an dem Socket

int **listen** (int socket, int backlog)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Nachdem der Socket *socket* mit socket() erzeugt wurde und mit bind() mit einem Namen verknüpft wurde, wird er mit dieser Funktion angewiesen, am Socket *socket* auf hereinkommende Verbindungen zu horchen. *backlog* ist das Maximum an hereinkommenden Verbindungsanforderungen, die zur weiteren Verarbeitung in die Warteschlange gestellt werden können.

listen() kann nur mit Sockets vom Typ `SOCK_STREAM` oder `SOCK_SEQPACKET` verwendet werden.

Gibt bei Erfolg `NULL` zurück oder einen negativen Fehlercode wenn die Funktion fehlschlägt. Dieser Fehlercode kann an die Funktion strerror() übergeben werden, um eine textuelle Beschreibung des Fehlers zu erhalten.

Siehe auch: accept_connect(), bind(), connect(), socket(), socket_get_status(), und strerror().

read (4.0.2 - 4.0.6 only)

Liest vom Socket

int **read** (int socket_des, string &buffer, int length)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **read()** liest vom Socket `socket_des`, der von der Funktion accept_connect() erzeugt wurde die Anzahl Bytes, die in dem Parameter `length` übergeben wurde in den Puffer `&buffer`. Um das Lesen zu Beenden, kann man \n, \t oder \0 benutzen. Gibt die Anzahl der gelesenen Bytes zurück.

Der optionale `type` ist eine benannte Konstante:

- PHP_BINARY_READ - benutzt die Systemfunktion **read()** (Standard in PHP >= 4.1.0)
- PHP_NORMAL_READ - Beendet das Lesen mit \n or \r. (Standard in PHP <= 4.0.6)

Siehe auch: accept_connect(), bind(), connect(), listen(), strerror(), socket_get_status() und write().

socket (4.0.2 - 4.0.6 only)

Erzeugt einen Socket (Endpunkt für die Datenübertragung)

int **socket** (int domain, int type, int protocol)

> **Warnung**
>
> Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Erzeugt eine Datenendeinrichtung zur Kommunikation (einen Socket) und gibt für diesen Socket einen Deskriptor zurück.

Der Parameter `domain` bestimmt die Domain. Momentan werden `AF_INET` und `AF_UNIX` unterstützt.

Der Parameter `type` wählt den Socket-Typ. Das ist entweder `SOCK_STREAM`, `SOCK_DGRAM`, `SOCK_SEQPACKET`, `SOCK_RAW`, `SOCK_RDM`, oder `SOCK_PACKET`.

`protocol` bestimmt das Protokoll.

Gibt bei Erfolg einen gültigen Socket-Deskriptor zurück oder einen negativen Fehlercode, falls ein Fehler auftritt.Dieser Fehlercode kann an die Funktion strerror() übergeben werden, um eine textuelle Beschreibung des Fehlers zu erhalten.

Um mehr Informationen über den Gebrauch der Funktion **socket()** und die Bedeutung der verschiedenen Parameter zu erhalten , sei auf die Unix man page socket (2) verwiesen.

Siehe auch: accept_connect(), bind(), connect(), listen(), strerror(), und socket_get_status().

strerror (4.0.2 - 4.0.6 only)

Gibt einen String zurück, der einen Socket-Fehler beschreibt

string **strerror** (int errno)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

strerror() nimmt als Parameter `errno` den Rückgabewert einer Socket-Funktion und gibt seinerseits den entsprechenden Erklärungstext aus. Das macht die Fehlersuche ein wenig einfacher. Zum Beispiel, anstatt eine Include-Datei des Systems zu durchsuchen, was der Fehlercode -111 bedeutet, kann man den Code an die Funktion **strerror()** übergeben und erhält Aufschluss über die internen Abläufe.

Beispiel 1. strerror() Beispiel:

```
<?php
if (($socket = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
   echo "socket() fehlgeschlagen: Grund: " . strerror ($socket) . "\n";
}

if (($ret = bind ($socket, '127.0.0.1', 80)) < 0) {
   echo "bind() fehlgeschlagen: Grund: " . strerror ($ret) . "\n";
}
?>
```

Die erwartete Ausgabe des obigen Beispiels (vorausgesetzt, das Skript wird nicht mit root-Benutzerrrechten ausgeführt) ist:

```
bind() fehlgeschlagen: Grund: Permission denied
```

Siehe auch: accept_connect(), bind(), connect(), listen(), socket(), und socket_get_status().

write (4.0.2 - 4.0.6 only)

Schreibt in den Socket

int **write** (int socket_des, string &buffer, int length)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Die Funktion **write()** schreibt aus dem Puffer `&buffer` die Anzahl Bytes, die mit `length` angegeben wurde, in den Socket mit dem Socket Deskriptor `socket_des`.

Siehe auch: accept_connect(), bind(), connect(), listen(), read(), strerror(), und socket_get_status().

XCV. String-Funktionen

Mit diesen Funktionen werden Strings (Zeichenketten) auf verschiedene Arten verändert. Spezielle Anwendungen können in den Abschnitten zu regulären Ausdrücken (regular expression) und der URL-Handhabung gefunden werden.

Informationen zum Funktionsweise von Strings (insbesondere beim Gebrauch von einfachen bzw. doppelten Anführungszeichen und Escape-Sequenzen) erhalten Sie unter dem Strings-Eintrag unter Typen des Manuals.

AddCSlashes (PHP 4 >= 4.0.0)

Stellt Zeichen eines Strings ein "\" voran (wie in C)

string **addcslashes** (string str, string charlist)

Es wird ein String zurück gegeben, in dem jedem Zeichen, das in *charlist* steht, ein "\" voran gestellt wurde. Teile wie `\n`, `\r` etc. werden wie in C übergangen, ASCII-Zeichen kleiner als 32 und grösser als 126 werden in ihren oktal-Wert umgesetzt. Seien Sie bei der Umsetzung von alphanumerischen Zeichen vorsichtig. Sie können in *charlist* einen Bereich wie "\0..\37" angeben. Dadurch wird allen ASCII-Zeichen von 0 bis 31 ein "\" voran gestellt.

Beispiel 1. addcslashes() Beispiel:

```
$escaped = addcslashes ($not_escaped, "\0..\37!@\177..\377");
```

Anmerkung: Diese Funktion ist ab Version PHP4b3-dev verfügbar.

Siehe auch stripcslashes(), stripslashes(), htmlspecialchars(), htmlspecialchars() und quotemeta().

AddSlashes (PHP 3, PHP 4 >= 4.0.0)

Stellt bestimmten Zeichen eines Strings einen Backslash "\" (Rückstrich) voran

string **addslashes** (string str)

Gibt einen String (Zeichenkette) zurück, in dem bestimmten Zeichen ein Backslash "\" voran gestellt wurde. Diese Funktion ist z.B. für Datenbankabfragen wichtig. Die behandelten Zeichen sind der einfache und der doppelte Anführungsstrich (`'` und `"`), der Rückstrich (`\`) selbst sowie NUL (das Null-Byte).

Siehe auch stripslashes(), htmlspecialchars() und quotemeta().

bin2hex (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Wandelt Binär-Daten in ihre hexadezimale Entsprechung um

string **bin2hex** (string str)

Es wird ein ASCII-String mit der hexadezimalen Wiedergabe des Parameters *str* zurück gegeben. Die Umwandlung geschieht byteweise mit dem höchsten Anteil (4 Bit, "high-nibble") zuerst.

chop (PHP 3, PHP 4 >= 4.0.0)

Entfernt Leerzeichen, Tabulatoren und Zeilenvorschübe am String-Ende

string **chop** (string str)

Gibt die in *str* enthaltene Zeichenkette ohne evtl. vorhandene Leerzeichen, Tabulatoren oder Zeilenvorschübe am String-Ende zurück.

Beispiel 1. chop()-Beispiel:

```
$string = "10 Leerzeichen am String-Ende          ";
echo chop($string)."?";

// Die Ausgabe lautet:
10 Leerzeichen am String-Ende?
```

Anmerkung: chop() unterscheidet sich von der Perl-Funktion *chop()*, die das letzte Zeichen eines Strings entfernt.

Siehe auch trim(), ltrim() und rtrim().

chr (PHP 3, PHP 4 >= 4.0.0)

Gibt ein einzelnes Zeichen zurück

string **chr** (int ascii)

Gibt einen 1-Zeichen-String zurück, der dem mit *ascii* angegebenem Zeichen entspricht.

Beispiel 1. chr() Beispiel:

```
$str .= chr (27); /* fügt an das Ende von $str ein Escape-Zeichen an */

/* Oft wird diese Funktionalität gebraucht */

$str = sprintf ("Dieser String endet mit einem Escape: %c", 27);
```

Diese Funktion ergänzt ord(). Siehe auch sprintf() mit dem Formatierungs-String `%c`.

chunk_split (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Zerlegt einen String in Teile gleicher Länge

string **chunk_split** (string string [, int chunklen [, string end]])

Kann für die Aufteilung eines Strings in kleinere Stücke verwendet werden. Dies ist z.B. hilfreich bei der Umwandlung von base64_encode-Ausgaben, die der RFC 2045 entsprechen müssen. Diese Funktion fügt nach per *chunklen* angegebene Zeichen (Vorgabe: alle 76 Zeichen) den mittels *end* angegebenen String (Vorgabe ist hierfür "\r\n") ein. Die Funktion gibt den neuen String zurück. Die Original-Zeichenkette bleibt unverändert.

Beispiel 1. chunk_split() Beispiel:

```
# Formatiere den String $daten in die RFC 2045 Schreibweise

$neuer_string = chunk_split (base64_encode($daten));
```

Diese Funktion arbeitet wesentlich schneller als ereg_replace().

Anmerkung: chunk_split() ist ab Version 3.0.6 vorhanden.

convert_cyr_string (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Setzt Strings von einem kyrillischen Zeichensatz in einen anderen um

string **convert_cyr_string** (string str, string from, string to)

Diese Funktion konvertiert den angegebenen String von einem kyrillischen Zeichensatz in einen anderen. Die Argumente *from* und *to* sind einzelne Zeichen die den Ausgangs- und Zielzeichensatz definieren. Die unterstützten Zeichensätze sind:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4 >= 4.0.0)

Gibt Informationen über die in einem String enthaltenen Zeichen zurück

mixed **count_chars** (string string [, mode])

Zählt die Häufigkeit des Vorkommens jedes einzelnen Byte-Wertes (0..255) in *string* und gibt sie auf verschiedene Arten zurück. Der optionale Parameter *mode* ist voreingestellt auf 0. In Abhängigkeit des *mode*-Wertes gibt **count_chars()** das folgende zurück:

- 0 - ein Array mit Byte-Werten für die Häufigkeit (auch 0-Vorkommen) jedes Zeichens.
- 1 - wie 0, aber nur Häufigkeiten grösser 0.
- 2 - wie 0, aber nur Häufigkeiten gleich 0.
- 3 - ein String der überhaupt vorkommenden Zeichen enthält.
- 4 - ein String der alle nicht vorkommenden Zeichen enthält.

Anmerkung: Diese Funktion existiert ab PHP 4.0.

crc32 (PHP 4)

Berechnet den polynomischen CRC32-Wert eines Strings

int **crc32** (string str)

Berechnet die zyklisch redundante polynomische Prüfsumme mit einer Länge von 32-Bit für `str`. Dies wird regulär für die Integritätsprüfung übermittelter Daten verwendet.

Siehe auch: md5().

crypt (PHP 3, PHP 4 >= 4.0.0)

DES-Verschlüsselung eines Strings

string **crypt** (string str [, string salt])

crypt() verschlüsselt einen String unter Verwendung der Standard-DES-Verschlüsselungs-Methode von Unix. Die zu übergebenden Argumente sind der zu verschlüsselnde String und optional ein 2-Zeichen-Salt-String, der die Schlüsselbasis bildet. Lesen Sie die Man-Pages ihres Unix-Systems für mehr Informationen zu den Crypt-Funktionen.

Ist kein Salt-Argument angegeben, wird es von PHP nach dem Zufalls-Prinzip erzeugt.

Einige Betriebssystem unterstützen mehr als eine Methode zur Verschlüsselung. So wird manchmal der DES- durch einen MD5-Algorithmus ersetzt. Der verwendete Schlüssel wird durch das Salt-Argument bestimmt. Zur Installations-Zeit untersucht PHP die vorhandenen Möglichkeiten und wird, abhängig vom Ergebnis dieser Prüfung, auch andere Schlüssel-Typen zulassen. Wird kein Salt unterstützt, erzeugt PHP per Voreinstellung einen 2-Zeichen DES-Salt, es sei denn auf ihrem System existiert MD5. Dann wird PHP einen zufälligen MD5-Salt generieren. PHP erzeugt eine Konstante namens CRYPT_SALT_LENGTH. Diese besagt, ob ihr System ein reguläres 2-Zeichen Salt oder das längere 12-Zeichen MD5-Salt unterstützt.

Wenn Sie das gegebene Salt benutzen, sollten Sie beachten, dass dieser Wert nur einmal bestimmt wird. Rufen Sie diese Funktion nun rekursiv auf, beeinflusst das nicht nur das Ergebnis, sondern unter Umständen auch die Sicherheit.

Die Standard-DES-Verschlüsselung **crypt()** enthält das Salt als erste 2 Zeichen der Ausgabe.

Auf Systemen, wo die crypt()-Funktion meherere Verschlüsselungen unterstützt, werden die folgenden Konstanten auf 0 oder 1 gesetzt, je nachdem, ob der entsprechende Typ verfügbar ist:

- CRYPT_STD_DES - Standard DES-Schlüssel mit 2-Zeichen Salt
- CRYPT_EXT_DES - Erweiterter DES-Schlüssel mit einem 9-Zeichen Salt
- CRYPT_MD5 - MD5-Schlüssel mit 12-Zeichen Salt, beginnend mit 1
- CRYPT_BLOWFISH - Erweiterter DES-Schlüssel, 16-Zeichen Salt, beginnend mit 2

Der veschlüsselte String kann nicht entschlüsselt werden, da **crypt()** eine Einweg-Verschlüsselung ist.

Siehe auch: md5().

echo (unknown)

Gibt einen oder mehrere Strings aus

echo (string arg1 [, string argn...])

Gibt alle als Parameter angegebenen Strings aus.

echo() ist keine Funktion, sonder ein Sprach-Konstrukt. Deshalb brauchen Sie keine Klammern zu verwenden.

Beispiel 1. echo()-Beispiel:

```
echo "Hallo Welt";

echo "Dieses
Beispiel
umfasst
mehrere
Zeilen.";
```

Anmerkung: Falls Sie mehr als einen auszugebenden Parameter angeben, dürfen Sie diese nicht in Klammern setzen.

Siehe auch: print(), printf() und flush().

explode (PHP 3, PHP 4 >= 4.0.0)

Zerteilt einen String anhand eines Trennzeichens

array **explode** (string separator, string string [, int limit])

Gibt ein Array aus Strings zurück. Jeder String ist ein Teil von *string*. Die Trennung erfolgt an der mit *separator* angegebenen Zeichenkette (ein oder mehrere Zeichen). Ist der Parameter *limit* angegeben, werden maximal *limit* Teile zurück gegeben. Das letzte Element enthält dann den ganzen Rest von *string*.

Beispiel 1. explode()-Beispiel:

```
$pizza = "Teil1 Teil2 Teil3 Teil4 Teil5 Teil6";
$pieces = explode (" ", $pizza);
```

Siehe auch split() und implode().

get_html_translation_table (PHP 4 >= 4.0.0)

Gibt die Übersetzungs-Tabelle zurück, die von htmlspecialchars() und htmlentities() benutzt wird

string **get_html_translation_table** (int table [, int quote_style])

get_html_translation_table() ermittelt die intern benutzte Übersetzungs-Tabelle, die von den Funktionen htmlspecialchars() und **htmlentities()** benutzt wird. Es gibt zwei neue Parameter (*HTML_ENTITIES* und *HTML_SPECIALCHARS*), die es ihnen ermöglichen, die gewünschte Tabelle auszuwählen und wie bei den htmlspecialchars() bzw. htmlentities()-Funktionen können Sie optional den Quote-Stil, mit dem Sie arbeiten, angeben. Voreingestellt ist der ENT_COMPAT-Modus. Vergleichen Sie dazu die Modus-Beschreibung unter htmlspecialchars().

Beispiel 1. Beispiel zur Übersetzungs-Tabelle

```
$trans = get_html_translation_table (HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr ($str, $trans);
```

Die `$encoded`-Variable wird jetzt "Hallo & <Frau> & Krämer". enthalten.

Mittels array_flip() können Sie die Richtung der Übersetzung umkehren.

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

Der Inhalt von `$original` wird "Hallo & <Frau> & Krämer" sein.

Anmerkung: Diese Funktion wurde in PHP 4.0 neu hinzu gefügt.

Siehe auch htmlspecialchars(), htmlentities(), strtr() und array_flip().

get_meta_tags (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Liest die Attribut-Inhalte aller Meta-Tags aus und gibt diese als Array zurück

array **get_meta_tags** (string filename [, int use_include_path])

Öffnet *filename* und untersucht diese Zeile für Zeile auf das Vorkommen von <meta>-Tags der Form

Beispiel 1. Meta-Tags Beispiel:

```
<meta name="author" content="Name">
<meta name="tags" content="PHP3-Dokumentation">
</head> <!-- Ende der Untersuchung -->
```

(Eine auf einem MAC erzeugte Datei kann nicht unter UNIX auf Meta-Tags untersucht werden, da die Zeilenenden anders sind.)

Der Wert der name-Eigenschaft wird Schlüssel, der Wert der content-Eigenschaft wird Wert des zurück gegebenen Arrays. Dadurch können Sie gewöhnliche Array-Funktionen nutzen, dieses zu durchlaufen oder einzelne Werte anzusprechen. Sonderzeichen innerhalb der name-Eigenschaft werden durch ´_´ ersetzt, der Rest wird in Kleinbuchstaben umgewandelt.

Ist *use_include_path* auf 1 gesetzt, wird PHP versuchen, die Datei anhand des Standard-Include-Pfades zu öffnen.

hebrev (PHP 3, PHP 4 >= 4.0.0)

Konvertiert (natürlichen) hebräischen Text in sichtbaren Text

string **hebrev** (string hebrew_text [, int max_chars_per_line])

Der optionale Parameter *max_chars_per_line* entspricht der Zeichenzahl pro Ausgabe-Zeile. Wortbrüche werden, soweit möglich, vermieden.

Siehe auch: hebrevc().

hebrevc (PHP 3, PHP 4 >= 4.0.0)

Konvertiert (natürlichen) hebräischen Text in sichtbaren Text incl. Anpassung von Zeilenumbrüchen

string **hebrevc** (string hebrew_text [, int max_chars_per_line])

Diese Funktion ist gleich hebrev(), mit dem Unterschied, dass Zeilenumbrüche (\n) zu "
\n" umgewandelt werden. Der optionale Parameter *max_chars_per_line* entspricht der Zeichenzahl pro Ausgabe-Zeile. Wortbrüche werden, soweit möglich, vermieden.

Siehe auch: hebrev().

htmlentities (PHP 3, PHP 4 >= 4.0.0)

Wandelt alle Sonderzeichen in entsprechende HTML-Codes um

string **htmlentities** (string string [, int quote_style])

Diese Funktion ist völlig identisch zu htmlspecialchars(), ausser dass alle Zeichen, die HTML-Code-Entsprechungen haben, in diese Codes umgewandelt werden. Wie bei htmlspecialchars() gibt es ein optionales zweites Argument, das die Behandlung von einfachen bzw. den doppelten Anführungszeichen beeinflusst. Im Modus ENT_COMPAT (default) werden nur die doppelten, im Modus ENT_QUOTES beide Arten und im Modus ENT_NOQUOTES keine Anführungszeichen umgewandelt.

Gegenwärtig wird der ISO-8859-1-Zeichensatz verwendet. Beachten Sie, dass das optionale 2. Argument erst ab PHP 3.0.17 bzw. PHP 4.0.3. verfügbar ist.

Siehe auch htmlspecialchars() und nl2br().

htmlspecialchars (PHP 3, PHP 4 >= 4.0.0)

Diese Funktion wandelt Sonderzeichen in HTML-Codes um

string **htmlspecialchars** (string string [, int quote_style])

Bestimmte Zeichen haben eine spezielle Bedeutung in HTML und sollten in HTML-Code dargestellt werden, um ihre Bedeutung zu behalten. Diese Funktion liefert einen String zurück, in dem diese Darstellung / Umsetzung teilweise durchgeführt wird. Die hier vorgenommenen Umsetzungen sind nützlich für die alltägliche Web-Programmierung. Sofern Sie eine vollständige Umsetzung der HTML-Zeichen benötigen, sollten Sie statt dessen htmlentities() verwenden.

Sinnvoll beim Schutz von benutzerabhängigen Texteingaben (z.B. in Gästebuch-Formularen). Das optionale 2. Argument (Quote-Stil) beeinflusst die Behandlung der einfachen und doppelten Anführungsstriche. Der Modus ENT_COMPAT (default) ist abwärtskompatibel und behandelt nur die doppelten, nicht aber die einfachen Anführungsstriche. ENT_QUOTES behandelt beide und ENT_NOQUOTES keine Art(en) von Anführungszeichen.

Folgende Umsetzungen werden vorgenommen:

- ’&’ (kaufmännisches und) wird zu ’&’
- ’"’ (doppelte Anführungszeichen) wird zu ’"’, wenn ENT_NOQUOTES nicht gesetzt ist
- ”’ (einfaches Anführungszeichen) wird zu ’'’, wenn ENT_QUOTES gesetzt ist
- ’<’ (kleiner als) wird zu ’<’
- ’>’ (grösser als) wird zu ’>’

Beispiel 1. htmlspecialchars()-Beispiel:

```
$neu = htmlspecialchars("<a href=’test’>Test</a>", ENT_QUOTES);
```

Beachten Sie, dass diese Funktion nur die o.a. Umwandlungen durchführt. Zur vollständigen Code-Umwandlung siehe htmlentities(). Beachten Sie auch, dass das optionale 2. Argument erst ab PHP 3.0.17 bzw. PHP 4.0.3. verfügbar ist.

Siehe auch htmlentities() und nl2br().

implode (PHP 3, PHP 4 >= 4.0.0)

Verbindet Array-Elemente zu einem String

string **implode** (string glue, array pieces)

Gibt einen String zurück, dessen Inhalt allen Array-Elementen in der fortlaufenden Reihenfolge ihres Vorkommens im Array entspricht. Die Stringteile sind durch den im Parameter glue enthaltenen String verbunden.

Beispiel 1. implode()-Beispiel:

```
$Trennung_durch_Doppelpunkt = implode (":", $array);
```

Siehe auch explode(), join() und split().

join (PHP 3, PHP 4 >= 4.0.0)

Join verbindet Array-Elemente zu einem String

string **join** (string glue, array pieces)

join() ist ein Alias von implode() (die Funktionalität ist also vollkommen identisch).

Siehe auch explode(), implode() und split().

levenshtein (PHP 3>= 3.0.17, PHP 4)

Berechnet den Levenshtein-Unterschied zweier Strings

int **levenshtein** (string str1, string str2)int **levenshtein** (string str1, string str2, int cost_ins, int cost_rep, int cost_del)int **levenshtein** (string str1, string str2, function cost)

Diese Funktion errechnet die Differenz zwischen den als Argumente übergebenen zwei Strings. Ist einer der Strings länger als die zulässigen 255 Zeichen, wird -1 zurück gegeben.

Die Levenshtein-Differenz ist definiert als die minimale Anzahl an Zeichen, die ersetzt, eingefügt oder gelöscht werden müssen um den *`str1`* nach *`str2`* umzusetzen. Die Komplexität des Algorithmus ist `O(m*n)`, wobei `n` und `m` der Länge der Strings *`str1`* und *`str2`* (viel besser als bei der Funktion similar_text(), wo sie O(max(n,m)**3) beträgt).

Siehe auch soundex(), similar_text() und metaphone().

ltrim (PHP 3, PHP 4 >= 4.0.0)

Entfernt führende Leerraum eines Strings

string **ltrim** (string str)

Es wird Leerraum ab Stringanfang entfernt und der um diesen gekürzte String zurück gegeben. Als Leerraum gelten die Zeichen "\n", "\r", "\t", "\v", "\0" und das Leerzeichen.

Siehe auch chop(), rtrim() und trim().

md5 (PHP 3, PHP 4 >= 4.0.0)

Errechnet den MD5-Code eines Strings

string **md5** (string str)

Berechnet den MD5-Code von `str` unter Verwendung des RSA Data Security, Inc. MD5 Message-Digest Algorithmus (http://www.faqs.org/rfcs/rfc1321.html).

Siehe auch: crc32().

metaphone (PHP 4 >= 4.0.0)

Berechnet den metaphone-Schlüssel eines Strings

string **metaphone** (string str)

Errechnet den metaphone-Schlüssel von `str`.

Genau wie soundex() berechnet metaphone den gleichen Schlüssel für ähnlich klingende Wörter. Die Metaphone-Funktion arbeitet genauer als soundex(), da sie die Grundregeln der englischen Aussprache kennt. Die durch metaphone erzeugten Schlüüsel sind von variabler Länge.

Der Metaphone-Algorithmus wurde entwickelt von Lawrence Philips <lphilips@verity.com> und ist beschrieben in ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

Anmerkung: Diese Funktion ist ab PHP 4.0 verfügbar.

nl2br (PHP 3, PHP 4 >= 4.0.0)

Fügt HTML-Zeilen-Umbrüche vor Zeilenumbrüche in Strings ein

string **nl2br** (string string)

Gibt den `string` mit einem eingefügten '<br />' vor dem Neue-Zeile-Zeichen zurück.

Siehe auch htmlspecialchars() und htmlentities() and wordwrap().

ord (PHP 3, PHP 4 >= 4.0.0)

Gibt den ASCII-Wert eines Zeichens zurück

int **ord** (string string)

Es wird der ASCII-Wert des ersten Zeichens von *string* zurück gegeben. Diese Funktion steht in Zusammenhang mit chr().

Beispiel 1. ord()-Beispiel:

```
if (ord ($str) == 10) {
    echo "Das erste Zeichen von \$str ist ein Zeilenvorschub.\n";
}
```

Siehe auch chr().

parse_str (PHP 3, PHP 4 >= 4.0.0)

Überträgt einen String in Variable

void **parse_str** (string str [, array arr])

Durchsucht *str* als ob er der Query-String einer URL-Übergabe via GET wäre und gibt die Werte Variableninhalte zurück. Ist der zweite Parameter *arr* gegeben, werden die Variablen-Werte in diesem Array als dessen Elemente gespeichert.

Anmerkung: Der optionale zweite Parameter wurde in PHP 4.0.3 hinzugefügt.

Beispiel 1. Der Gebrauch von parse_str:()

```
$str = "erster=nichts&zweiter[]=etwas&zweiter[]=nochwas";
parse_str($str);
echo $erster;     /* gibt "nichts" aus */
echo $zweiter[0]; /* gibt "etwas" aus */
echo $zweiter[1]; /* gibt "nochwas" aus */
```

print (unknown)

Ausgabe eines Strings

print (string arg)

Gibt `arg` aus.

Siehe auch echo(), printf() und flush().

printf (PHP 3, PHP 4 >= 4.0.0)

Gibt einen formatierten String aus

int **printf** (string format [, mixed args])

Erzeugt eine Ausgabe abhängig von `format`. `Format` wird in der Dokumentation zu sprintf() näher beschrieben.

Siehe auch print(), sprintf(), sscanf(), fscanf() und flush().

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Konvertiert einen "quoted-printable"-String in einen 8-Bit-String

string **quoted_printable_decode** (string str)

Diese Funktion gibt einen 8-Bit-String zurück, der der Konvertierung aus einem "quoted-printable"-String entspricht. Diese Funktion ist ähnlich wie imap_qprint() bis auf die Tatsache, dass sie kein IMAP-Module erfordert.

QuoteMeta (PHP 3, PHP 4 >= 4.0.0)

Quoten von Meta-Zeichen

string **quotemeta** (string str)

Der Inhalt von `str` wird mit einem Rückstrich ("Backslash", \) vor jedem Vorkommen von

```
. \\ + * ? [ ^ ] ( $ )
```

versehen ("quoten")und zurück gegeben.

Siehe auch addslashes(), htmlentities(), htmlspecialchars(), nl2br() und stripslashes().

rtrim (PHP 3, PHP 4 >= 4.0.0)

Entfernt Leerzeichen am String-Ende

string **rtrim** (string str)

Gibt den als Argument übergebenen String ohne abschließende Leerzeichen (auch "Neue-Zeile-"Steuerzeichen werden entfernt). Dies ist ein Alias zu chop().

Beispiel 1. rtrim()-Beispiel:

```
$getrimmt = rtrim ($line);
```

Siehe auch chop(), ltrim() und trim().

sscanf (PHP 4)

Übertragt die Eingaben eines Strings in ein angegebenes Format

mixed **sscanf** (string str, string format [, string var1])

Die Funktion **sscanf()** ist die Eingabe analog zu printf(). **sscanf()** liest den *str*-String und interpretiert ihn Anlehnung an die angegebenen *format* Parameter. Werden nur 2 Parameter angegeben, werden die analysierten Werte als Array zurück gegeben.

Beispiel 1. sscanf()-Beispiel:

```
// Ermittlung der Serien-Nr.
$serial = sscanf("SN/2350001","SN/%d");
// und des Herstellungsdatums
$mandate = "Januar 01 2000";
list($monat, $tag, $jahr) = sscanf($mandate,"%s %d %d");
echo "Das Teil $serial wurde hergestellt am: $jahr-".substr($monat,0,3)."-$tag\n";
```

Werden optionale Parameter übergeben, gibt die Funktion die Anzahl der ermittelten Werte zurück. Die optionalen Parameter müssen refenziert übergeben werden.

Beispiel 2. sscanf() - Gebrauch optionaler Parameter

```
// Ermittlung der Autoren-Info und Erzeugung eines DocBook-Eintrages
$auth = "24\tLewis Carroll";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<author id='$id'>
      <firstname>$first</firstname>
      <surname>$last</surname>
      </author>\n";
```

Siehe auch: fscanf(), printf() und sprintf().

setlocale (PHP 3, PHP 4 >= 4.0.0)

Setzt lokale Informationen

string **setlocale** (string category, string locale)

Der String *category* enthält die Kategorie der zu ändernden Lokalitäts-Informationen:

- LC_ALL - alle folgenden Werte
- LC_COLLATE - für String-Vergleiche - z.Zt. noch nicht in PHP implementiert
- LC_CTYPE - betrifft Klassifizierungen und Umwandlung, z.B. bei strtoupper()
- LC_MONETARY - betrifft localeconv() - z.Zt. noch nicht in PHP implementiert
- LC_NUMERIC - für das Dezimal-Trennzeichen
- LC_TIME - betrifft Zeit und Datums-Formatierungen durch strftime()

Ist *locale* als Leer-String definiert, also `""`, werden die Werte an Hand der entsprechenden Umgebungs-Variablen gleichen Namens oder von "LANG" bestimmt.

Ist *locale* `NULL` oder `"0"`, werden keine Einstellungen verändert sondern als Ergebnis zurück gegeben.

setlocale() gibt die neue Einstellung oder `FALSE` (wenn die locale-Funktionalität auf der System-Plattform nicht unterstützt wird oder der Kategorie-Wert ungültig ist) zurück. Ein ungültiger Kategorie-Name erzeugt auch noch eine Warnmeldung.

similar_text (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Berechnet die Ähnlichkeit zweier Zeichenketten

int **similar_text** (string first, string second [, double percent])

Die Berechnung der Ähnlichkeit zweier Strings erfolgt nach der in "Oliver" [1993] beschreibenen Methode. Beachten Sie, dass diese Implementierung keinen Stack verwendet wie in Olivers Pseude-Code, dafür aber rekursive Aufrufe, die die Performance erhöhen können (aber nicht müssen). Beachten Sie auch, dass die Komplexität dieses Algorithmus O(N**3) beträgt, wobei N die Länge der längsten Zeichenkette ist.

Falls Sie als Referenz ein drittes Argument angeben, wird **similar_text()** ihnen die Ähnlichkeit als Prozentwert errechnen. Es wird die Anzahl der in beiden Strings übereinstimmenden Zeichen zurück gegeben.

soundex (PHP 3, PHP 4 >= 4.0.0)

Berechnet die Laut-Ähnlichkeit eines Strings

string **soundex** (string str)

Errechnet den Wert der Laut-Ähnlichkeit von *str*.

Soundex-Werte haben die Eigenschaft, dass ähnlich ausgesprochene Wörter den gleichen Soundex-Wert erzeugen. Dies kann zur Suche in Datenbanken verwendet werden, wenn Sie zwar die Aussprache aber nicht die genaue Schreibweise kennen. **soundex()** liefert einen String aus 4 Zeichen, beginnend mit einem Buchstaben zurück.

Diese besondere **soundex()**-Funktion ist u.a. beschrieben in "The Art Of Computer Programming, vol. 3: Sorting And Searching", von Donald Knuth, Addison-Wesley-Verlag (1973), Seiten 391 bis 392.

Beispiel 1. Soundex-Beispiel:

```
soundex ("Euler") == soundex ("Ellery") == 'E460';
soundex ("Gauss") == soundex ("Ghosh") == 'G200';
soundex ("Knuth") == soundex ("Kant") == 'H416';
soundex ("Lloyd") == soundex ("Ladd") == 'L300';
soundex ("Lukasiewicz") == soundex ("Lissajous") == 'L222';
```

sprintf (PHP 3, PHP 4 >= 4.0.0)

Gibt einen formatierten String zurück

string **sprintf** (string format [, mixed args])

Gibt einen an Hand des Formatierungs-Strings `format` gebildeten String zurück.

Der Formatierungs-String wird aus keiner oder mehreren Anweisungen gebildet: Gewöhnliche Zeichen (ausser `%`), die direkt in das Ergebnis übertragen werden und *Umsetzungs-Anweisungen*. Jede dieser Anweisungen resultiert ihre eigene Auswirkung. Diese Anweisungen gelten sowohl für die **sprintf()**- als auch für die printf()-Funktion.

Jede der Formatierugs-Anweisungen besteht aus einem Prozent-Zeichen gefolgt von einem oder mehrereren der folgenden Elemente (in alphabetischer Reihenfolge):

1. Einem optionalen *Füll-Zeichen*, das den String bis rechts auffüllt. Dies kann ein Leerzeichen oder eine `0` (das Null-Zeichen) sein. Als Füllzeichen voreingestellt ist das Leerzeichen. Andere Füllzeichen können durch Voranstellung eines einfachen Anführungszeichens (`'`) definiert werden. Beachten Sie die folgenden Beispiele.
2. Einer optionalen Angabe der *Ausrichtung*, ob die Ausgabe also nach links oder nach rechts (default) ausgerichtet werden soll. Um den Text links auszurichten müssen Sie das Zeichen `-` angeben.
3. Optional einem Wert für die *Längen-Angabe*, also der Wert für die mindestenst zurückzugebenden Zeichen.
4. Optional einem Wert für die Anzahl der auszugebenden *Nachkomma-Stellen*. Dieser Wert hat nur Einfluss auf Werte vom Typ "double" (eine weitere hilfreiche Funktion für die Formatierung von Zahlen ist number_format()).
5. Einer *Typ-Angabe* welche den Typ enthält, als der das Argument angesehen werden soll. Mögliche Angaben sind:
   `%` - ein Prozent-Zeichen ("%"). Es ist kein Argument erforderlich.

`b` - das Argument wird als Integer angesehen und als Binär-Wert ausgegeben.
`c` - das Argument wird als Integer angesehen und das entsprechende ASCII-Zeichen wird ausgegeben.
`d` - das Argument wird als Integer angesehen und ein Dezimalwert ausgegeben.
`f` - das Argument wird als "double" angesehen und eine Fließkomma-Zahl ausgegeben.
`o` - das Argument wird als Integer angesehen und und als Oktalwert ausgegeben.
`s` - das Argument wird als String angesehen und als solcher ausgegeben.
`x` - das Argument wird als Integer angesehen und als Hexadezimal-Wert ausgegeben (mit Kleinbuchstaben).
`X` - das Argument wird als Integer angesehen und als Hexadezimal-Wert ausgegeben (mit Großbuchstaben).

Siehe auch: printf(), sscanf(), fscanf() und number_format().

Beispiel 1. sprintf(): Null-gefüllte Integer-Werte

```
$isodatum = sprintf ("%04d-%02d-%02d", $jahr, $monat, $tag);
```

Beispiel 2. sprintf(): Integer- zu Binärwert

```
$wert = 155;
$binaer = sprintf ("%b", $wert);
echo ("<br> $binaer <br>");
// gibt 10011011 aus
```

Beispiel 3. sprintf(): Währungsformatierung

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money wird "123.1" ausgeben
$formatiert = sprintf ("%01.2f", $money);
// echo $formatiert wird "123.10" ausgeben
```

strcasecmp (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Binärer Vergleich von Zeichenketten

int **strcasecmp** (string str1, string str2)

Ist *`str1`* kleiner als *`str2`* wird < 0, ist *`str1`* grösser als *`str2`* wird > 0 und bei Gleichheit wird 0 zurück gegeben.

Beispiel 1. strcasecmp()-Beispiel:

```
$var1 = "Hallo";
$var2 = "hallo";
if (!strcasecmp($var1, $var2) ) {
    echo '$var1 ist wie $var2 bei schreibweise-unabhängigem Vergleich';
```

```
}
```

Beachten Sie, dass nicht zwischen Groß- und Kleinschreibung unterschieden wird.

Siehe auch ereg(), strcmp(), substr(), stristr() und strstr().

strchr (PHP 3, PHP 4 >= 4.0.0)

Findet das erste Vorkommen eines Zeichens

string **strchr** (string haystack, string needle)

Diese Funktion ist ein Alias zu strstr() und funktioniert genau wie dort beschrieben.

strcmp (PHP 3, PHP 4 >= 4.0.0)

Binärer Vergleich zweier Strings

int **strcmp** (string str1, string str2)

Ist *str1* kleiner als *str2* wird < 0, ist *str1* grösser als *str2* wird > 0 und bei Gleichheit wird 0 zurück gegeben.

Beachten Sie, dass zwischen Groß- und Kleinschreibung unterschieden wird.

Siehe auch ereg(), strcasecmp(), substr(), stristr(), strncmp() und strstr().

strcspn (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Ermittelt die Anzahl der nicht übereinstimmenden Zeichen

int **strcspn** (string str1, string str2)

Die Länge des Strings *str1* bis zum ersten Erscheinen eines Buchstabens (aus *str1*) in *str2* wird zurück gegeben.

Beispiel 1. strcspn()-Beispiel:

```
$s1 = "dasrktuvs";
$s2 = "iklp";                  // "k" ist der "Kasus-Knacktus"
$wert = strcspn ($s1, $s2);
echo "<br> $wert <br>";  // gibt also 4 zurück
```

Siehe auch strspn().

strip_tags (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Entfernt HTML- und PHP-Tags aus einem String

string **strip_tags** (string str [, string allowable_tags])

Diese Funktion versucht, alle HTML- und PHP-Tags aus `str` zu entfernen. Bei unvollständigen oder sinnlosen Tags werden Fehler ausgegeben. Es wird die gleiche Art der Tag-Engine wie bei der fgetss()-Funktion benutzt.

Sie können den optionalen Parameter `allowable_tags` benutzen, um nicht zu entfernende Tags zu definieren.

Anmerkung: `Allowable_tags` steht ab PHP 3.0.13 bzw. PHP4B3 zur Verfügung.

StripCSlashes (PHP 4 >= 4.0.0)

Umkehrung / Gegenstück zu addcslashes

string **stripcslashes** (string str)

Gibt einen String ohne "\" zurück. Wie in C werden `\n`, `\r` ..., oktal- und hexadezimal-Vorkommen erkannt.

Anmerkung: Verfügbar seit PHP4b3-dev.

Siehe auch addcslashes().

StripSlashes (PHP 3, PHP 4 >= 4.0.0)

Umkehrung / Gegenstück zu addslashes

string **stripslashes** (string str)

Gibt einen String ohne evtl. vorhandene "\" (Backslash) zurück (`\'` wird zu `'` usw.). Doppelte Rückstriche ("\\") werden zu "\" umgesetzt.

Siehe auch addslashes().

stristr (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Wie strstr() ohne Rücksicht auf Groß- bzw. Kleinschreibung

string **stristr** (string haystack, string needle)

Findet alle Vorkommen von *needle* innerhalb von *haystack* vom Anfang bis zum Ende (ohne Rücksicht auf die Schreibweise beider Parameter).

Wird *needle* nicht gefunden, wird FALSE zurück gegeben.

Ist *needle* kein String, wird er in einen Integer-Wert umgesetzt. Dies ASCII-Entsprechung dieses Wertes wird dann als zu suchendes Zeichen angesehen.

Siehe auch strchr(), strrchr(), substr() und ereg().

strlen (PHP 3, PHP 4 >= 4.0.0)

Ermitteln der String-Länge

int **strlen** (string str)

Gibt die Länge der Zeichenkette *string* zurück.

strnatcmp (PHP 4 >= 4.0.0)

String-Vergleich unter Verwendung einer "natürlichen Ordnung"

int **strnatcmp** (string str1, string str2)

Diese Funktion implemetiert einen Vergleich, der alphanumerische Zeichen "menschlich" einordnet (auch "natürliche Ordnung" genannt). Der Unterschied zwischen dieser und der regulären String-Sortierung eines Computer (unter Verwendung von strcmp()) wird in dem folgenden Beispiel gezeigt:

```
$arr1 = $arr2 = array ("img12.png","img10.png","img2.png","img1.png");
echo "Normaler String-Vergleich\n";
usort($arr1,"strcmp");
print_r($arr1);
echo "\nVergleich mittels natürlicher Ordnung\n";
usort($arr2,"strnatcmp");
print_r($arr2);
```

Der o.a. Code erzeugt die folgende Ausgabe:

```
Normaler String-Vergleich
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Vergleich mittels natürlicher Ordnung
Array
```

```
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Für weitere Infos siehe Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) Seite.

Ähnlich anderen String-Vergleichs-Funktionen gibt diese < 0 zurück, wenn `str1` kleiner ist als `str2` bzw. > 0, wenn `str1` größer ist als `str2` oder 0, wenn sie gleich sind.

Dieser Vergleich unterscheidet Groß- und Kleinschreibung.

Siehe auch ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strnatcasecmp(), strstr(), natsort() und natcasesort() .

strnatcasecmp (PHP 4 >= 4.0.0)

String-Vergleich "natürlicher Ordnung" ohne Unterscheidung der Schreibweise

int **strnatcasecmp** (string str1, string str2)

Diese Funktion implementiert einen String-Vergleich wie unter strnatcmp() mit dem Unterschied, dass hier nicht zwischen Groß- und Kleinschreibung unterschieden wird. Für weitere Infos siehe Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) Seite.

Ähnlich anderen String-Vergleichs-Funktionen gibt diese < 0 zurück, wenn `str1` kleiner ist als `str2` bzw. > 0, wenn `str1` größer ist als `str2` oder 0, wenn sie gleich sind.

Siehe auch ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strnatcmp() und strstr().

strncmp (PHP 4 >= 4.0.0)

Binary safe string comparison of the first n characters

int **strncmp** (string str1, string str2, int len)

Diese Funktion ist ähnlich strcmp() nur mit dem Unterschied, dass sie den maximalen Wert der Zeichen (`len`) angeben können, der verglichen werden soll. Ist einer der Strings kürzer als (`len`) wird dessen Länge für den Vergleich angewendet.

Gibt < 0 zurück, wenn `str1` kleiner ist als `str2` bzw. > 0, wenn `str1` größer ist als `str2` oder 0, wenn sie gleich sind.

Dieser Vergleich unterscheidet Klein- und Großschreibung.

Siehe auch ereg(), strcasecmp(), substr(), stristr(), strcmp() und strstr().

str_pad (PHP 4)

Erweitert einen String auf eine bestimmte Länge unter Verwendung eines anderen Strings

string **str_pad** (string input, int pad_length [, string pad_string [, int pad_type]])

Mit dieser Funktion erweitern Sie den String *input* auf der linken, rechten oder beiden Seiten auf die mit dem optionalen Parameter *pad_type* angegebene Länge. Wird das optionale Argument *pad_string* nicht angegeben, erfolgt die Erweiterung mit Leerzeichen, ansonsten wird hierzu der Inhalt von *pad_string* verwendet.

Das optionale Argument *pad_type* kann STR_PAD_RIGHT, STR_PAD_LEFT oder STR_PAD_BOTH sein. Wird *pad_type* nicht angegeben, so wird per default von STR_PAD_RIGHT ausgegangen.

Ist der Wert von *pad_length* negativ oder kleiner als die Länge des Input-Strings, wird keine Erweiterung vorgenommen.

Beispiel 1. str_pad()-Beispiel:

```
$input = "Alien";
print str_pad ($input, 10);                      // Ergebnis "Alien     "
print str_pad ($input, 10, "-=", STR_PAD_LEFT); // Ergebnis "-=-=-Alien"
print str_pad ($input, 10, "_", STR_PAD_BOTH);  // Ergebnis "__Alien___"
```

strpos (PHP 3, PHP 4 >= 4.0.0)

Ermitteln des ersten Vorkommens innerhalb eines Strings

int **strpos** (string haystack, string needle [, int offset])

Gibt als numerischen Wert die Position des ersten Vorkommens von *needle* innerhalb der Zeichenkette *haystack* zurück. Anders als bei strrpos() kann diese Funktion eine komplette Zeichenkette unterstützen.

Wurde *needle* nicht gefunden, wird FALSE zurück gegeben.

Anmerkung: Die Meldungen "Zeichen bei Position 0 gefunden (character found at position 0)" und "Zeichen nicht gefunden (character not found)" werden oft falsch ausgelegt. Hier die Unterscheidung:

```
// in PHP 4.0b3 und danach:
$pos = strpos ($mystring, "b");
if ($pos === false) { // Achtung: 3 Gleichheits-Zeichen
    // nicht gefunden...
}

// in Versionen vor 4.0b3:
$pos = strpos ($mystring, "b");
if (is_string ($pos) && !$pos) {
    // nicht gefunden...
}
```

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Der optionale Parameter *offset* ermöglicht es, den Startwert für die Suche nach *needle* innerhalb von *haystack* anzugeben. Die zurück gegebene Positions-Angabe ist dann relativ zum Anfang von haystack.

Siehe auch strrpos(), strrchr(), substr(), stristr() und strstr().

strrchr (PHP 3, PHP 4 >= 4.0.0)

Findet das letzte Vorkommen eines Zeichens innerhalb einer Zeichenkette

string **strrchr** (string haystack, string needle)

Diese Funktion gibt den letzten Teil des Strings *haystack* zurück, der ab dem letzten Vorkommen von *needle* steht (bis zum Ende).

Wird *needle* nicht gefunden, wird FALSE zurück gegeben.

Enthält *needle* mehr als ein Zeichen, wird nur das erste Zeichen genommen.

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Beispiel 1. strrchr()-Beispiel:

```
// hole das letzte Verzeichnis aus $PATH
$dir = substr (strrchr ($PATH, ":"), 1);

// hole alles nach dem letzten "Neue-Zeile"
$text = "Zeile 1\nZeile 2\nZeile 3";
$last = substr (strrchr ($text, 10), 1 );
```

Siehe auch substr(), stristr() und strstr().

str_repeat (PHP 4 >= 4.0.0)

Wiederholung eine String-Ausgabe

string **str_repeat** (string input, int multiplier)

Die Ausgabe von *input_str* wird *multiplier* mal wiederholt. *Multiplier* muss grösser als 0 sein.

Beispiel 1. str_repeat()-Beispiel:

```
echo str_repeat ("-=", 10);
```

Es wird "-=-=-=-=-=-=-=-=-=-=" ausgegeben.

Anmerkung: Verfügbar ab PHP 4.0.

strrev (PHP 3, PHP 4 >= 4.0.0)

Umdrehen eines Strings

string **strrev** (string string)

Dreht die Schreibweise von *string* um.

strrpos (PHP 3, PHP 4 >= 4.0.0)

Findet letztes Vorkommen eines Zeichens in einem String

int **strrpos** (string haystack, char needle)

Der Wert des letzen Vorkommens von *needle* im String *haystack* wird ermittelt. *Needle* darf nur ein einzelnes Zeichen sein. Wird als *needle* eine Zeichenkette übergeben, wird nur dessen erster Buchstabe ausgewertet.

Wurde *needle* nicht gefunden, wird FALSE zurück gegeben.

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Siehe auch strpos(), strrchr(), substr(), stristr() und strstr().

strspn (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Ermittelt die Länge der übereinstimmenden Zeichen

int **strspn** (string str1, string str2)

Ermittelt die Anzahl der Zeichen am Anfang von *str1*, die aus dem in *str2* übergebenen Zeichenvorrat stammen.

So ergibt z.B.

```
strspn("42 ist die Antwort, was ist die Frage ...","1234567890");
```

als Ergebnis 2.

Siehe auch strcspn().

strstr (PHP 3, PHP 4 >= 4.0.0)

Findet das erste Vorkommen eines Strings

string **strstr** (string haystack, string needle)

Es wird alles von *haystack* zurück gegeben ab der ersten Fundstelle von *needle* bis zum Ende.

Falls *needle* nicht gefunden wird, ist das Ergebnis FALSE.

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Diese Funktion unterscheidet zwischen Groß- und Kleinschreibung. Ist keine Unterscheidung gewünscht / erforderlich, sollten Sie stristr() verwenden.

Beispiel 1. strstr()-Beispiel:

```
$email = 'sterling@designmultimedia.com';
$domain = strstr ($email, '@');
print $domain; // gibt @designmultimedia.com zurück
```

Siehe auch stristr(), strrchr(), substr() und ereg().

strtok (PHP 3, PHP 4 >= 4.0.0)

Zerlegt einen String

string **strtok** (string arg1, string arg2)

strtok() wird benutzt, um einen String (*arg1*) zu teilen. Den Satz "Das ist ein Beispiel" können Sie in seine einzelnen Wörter zerlegen, in dem Sie das Leerzeichen als Trennzeichen (*arg2*) angeben.

Beispiel 1. strtok()-Beispiel:

```
$string = "Das ist noch ein Beispiel";
$tok = strtok ($string," ");
while ($tok) {
    echo "Wort=$tok<br>";
    $tok = strtok (" ");
}
```

Beachten Sie, dass der zu zerlegende String nur beim ersten Aufruf von **strtok()** angegeben werden muss. Die nachfolgenden Aufrufe brauchen nur das Trennzeichen als Parameter. Falls Sie eine neue Zeichenkette zerlegen wollen, rufen Sie **strtok()** wieder mit dem Parameter *arg1* auf. Das

initialisiert die Funktion. Sie können auch meherere Trennzeichen in *arg2* angeben. Die Trennung erfolgt beim ersten Auftreten eines beliebigen Zeichens aus *arg2*.

Beachten Sie, dass die Teile auch "0" sein können. Dies entspricht in bedingten Ausdrücken dem Wert FALSE.

Siehe auch split() und explode().

strtolower (PHP 3, PHP 4 >= 4.0.0)

Setzt einen String in Kleinbuchstaben um

string **strtolower** (string str)

Gibt *string* zurück, in dem alle Buchstaben in Kleinbuchstaben umgewandelt wurden.

Beachten Sie, dass 'Buchstaben' vom Wert locale abhängig ist. Ist z.B. die Voreinstellung für locale "C", werden keine Umlaute (Ä, Ü, Ö) umgewandelt.

Beispiel 1. strtolower()-Beispiel:

```
$str = "Dieter B. (M.T.) Ist Ein Schaf und Er LIEBT Sich So";
$str = strtolower($str);
print $str; // Gibt "dieter b. (m.t.) ist ein schaf und er liebt sich
            // so" aus.
```

Siehe auch strtoupper() und ucfirst().

strtoupper (PHP 3, PHP 4 >= 4.0.0)

Setzt einen String in Großbuchstaben um

string **strtoupper** (string string)

Gibt *string* zurück, in dem alle Buchstaben in Großbuchstaben umgewandelt wurden.

Beachten Sie, dass 'Buchstaben' vom Wert locale abhängig ist. Ist z.B. die Voreinstellung für locale "C", werden keine Umlaute (ä, ü, ö) umgewandelt.

Beispiel 1. strtoupper()-Beispiel:

```
$str = "Dieter B. (M.T.) Ist Ein Schaf und Er LIEBT Sich So";
$str = strtoupper($str);
print $str; // Gibt "DIETER B. (M.T.) IST EIN SCHAF UND ER LIEBT SICH
            // SO" aus.
```

Siehe auch strtolower() und ucfirst().

str_replace (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ersetzt alle Vorkommen eines Strings in einem anderen String

string **str_replace** (string needle, string str, string haystack)

Diese Funktion ersetzt alle Vorkommen von *needle* innerhalb der Zeichenkette *haystack* durch den String *str*. Falls Sie keine ausgefallenen String-Ersetzungen brauchen, sollten Sie immer dieser Funktion den Vorzug vor ereg_replace() geben.

Beispiel 1. str_replace()-Beispiel:

```
$bodytag = str_replace ("%body%", "black", "<body text=%body%>");
```

Diese Funktion beruht auf einem Binär-Vergleich.

Anmerkung: str_replace() ist seit PHP 3.0.6 verfügbar, war aber bis PHP 3.0.8 fehlerhaft.

Siehe auch ereg_replace() und strtr().

strtr (PHP 3, PHP 4 >= 4.0.0)

Tauscht bestimmte Zeichen aus

string **strtr** (string str, string from, string to)

Diese Funktion bearbeitet *str*, in dem alle Vorkommen von Zeichen aus *from* in die entsprechenden Zeichen aus *to* umgesetzt werden. Zurück gegeben wird der bearbeitete String.

Sind *from* und *to* von unterschiedlicher Länge werden die überzähligen Zeichen ignoriert.

Beispiel 1. strtr()-Beispiel:

```
$addr = strtr($addr, "äåö", "aao");
```

strtr() kann auch mit nur 2 Argumenten aufgerufen werden. Die Funktionsweise ist dann anders: *From* muss dann ein Array sein, welches String -> String Paare enthält, die innerhalb des Quellstrings ausgetauscht werden sollen. *Strtr* sucht immer erst die längste Entsprechung und wird keine Auswechselungen vornehmen, die bereits bearbeitet wurden.

Beispiele dazu:

```
$trans = array ("hallo" => "hi", "hi" => "hallo");
echo strtr("hi alle, ich sage hallo", $trans) . "\n";
```

Die Ausgabe ist: "hallo alle, ich sage hi",

Anmerkung: Die Möglichkeit mit 2 Parametern wurde in PHP 4.0 hinzugefügt.

Siehe auch ereg_replace().

substr (PHP 3, PHP 4 >= 4.0.0)

Gibt einen Teilstring zurück

string **substr** (string string, int start [, int length])

substr() gibt den Teil von *string* zurück, der durch die *start* und *length* Parameter definiert wurde.

Wenn *start* positiv ist, beginnt der zurück gegebene String an der *start*-Position von *string*, angefangen bei NULL. So ist z.B. im String 'abcdef' das Zeichen an der Position 0 gleich 'a', das Zeichen an der Position 2 ist 'c' usw.

Beispiele:

```
$rest = substr ("abcdef", 1);    // gibt "bcdef" zurück
$rest = substr ("abcdef", 1, 3); // gibt "bcd" zurück
```

Ist *start* negativ, gibt er den Beginn-Wert ab Ende des Strings an (also rückwärts gezählt).

Beispiele:

```
$rest = substr ("abcdef", -1);     // gibt "f" zurück
$rest = substr ("abcdef", -2);     // gibt "ef" zurück
$rest = substr ("abcdef", -3,  1); // gibt "d" zurück
$rest = substr ("abcdef",  1, -1); // gibt "bcde" zurück
$rest = substr ("abcdef",  4, -4); // gibt "" zurück (nichts)
$rest = substr ("abcdef",  7,  1); // gibt "" zurück (nichts)
$rest = substr ("abcdef",  7, -4); // gibt "" zurück (nichts)
$rest = substr ("abcdef",  4,  4); // gibt "ef" zurück
$rest = substr ("abcdef", -2,  4); // gibt "ef" zurück
$rest = substr ("abcdef", -5, -4); // gibt "b" zurück
```

Siehe auch strrchr() und ereg().

substr_count (PHP 4 >= 4.0.0)

Ermittelt die Häufigkeit des Vorkommens einer Zeichenkette in einem String

int **substr_count** (string haystrack, string needle)

Die Funktion **substr_count()** ermittelt, wie oft *needle* in dem String *haystack* vorkommt.

Beispiel 1. substr_count()-Beispiel:

```
print substr_count("Dieser Test ist Mist", "ist"); // ergibt 2
```

Diese Funktion steht ab PHP 4.0.0 zur Verfügung.

substr_replace (PHP 4 >= 4.0.0)

Ersetzt Text in einer Zeichenkette

string **substr_replace** (string string, string replacement, int start [, int length])

substr_replace() ersetzt den Teil von *string*, der durch die Parameter *start* und (optional) *length* begrenzt wird mit der in *replacement* angegeben Zeichnekette. Das Ergebnis wird zurück gegeben.

Ist *start* positiv, beginnt der Austausch am Offset *start* innerhalb von *string*.

Ist *start* negativ, wird vom Ende der Zeichenkette *string* bis zum Wert von *start* rückwärts gezählt und dort mit dem Austausch begonnen.

Ist *length* angegeben und positive, stellt dieser Parameter die Länge des auszuwechselnden Teilstrings dar. Ist *length* angegeben und negativ, gibt er die Zeichenzahl an um die ab Ende von *string* rückwärts gezählt wird. Bis zu dieser Stelle erfolgt dann der Austausch. Ist *length* nicht angegeben, wird unterstellt, dass Sie die Ersetzung bis zum String-Ende (strlen(*string*)) bezwecken.

Beispiel 1. substr_replace()-Beispiel:

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* Diese 2 Beispiele ersetzen alles in $var mit 'bob'. */
echo substr_replace ($var, 'bob', 0) . "<br>\n";
echo substr_replace ($var, 'bob', 0, strlen ($var)) . "<br>\n";

/* Fügt 'bob' am Anfang von $var ein . */
echo substr_replace ($var, 'bob', 0, 0) . "<br>\n";

/* Die nächsten beiden ersetzen 'MNRPQR' in $var durch 'bob'. */
echo substr_replace ($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace ($var, 'bob', -7, -1) . "<br>\n";

/* Löscht 'MNRPQR' aus $var. */
echo substr_replace ($var, '', 10, -1) . "<br>\n";
?>
```

Sieehe auch str_replace() und substr().

Anmerkung: substr_replace() existiert ab PHP 4.0.

trim (PHP 3, PHP 4 >= 4.0.0)

Entfernt überflüssige Zeichen ("whitespace") am Anfang und Ende eines Strings

string **trim** (string str)

Diese Funktion entfernt Leerraum am Anfang und Ende einer Zeichenkette. Der gekürzte String wird zurück gegeben. Als überflüssige Zeichen werden z.Zt. angesehen: "\n", "\r", "\t", "\v", "\0" und normale Leerzeichen.

Siehe auch chop(), ltrim(), und rtrim().

ucfirst (PHP 3, PHP 4 >= 4.0.0)

Setzt das erste Zeichen eines Strings als Großbuchstabe um

string **ucfirst** (string str)

Setzt das erste Zeichen von *str* in einen Großbuchstaben um, wenn es ein Zeichen des Alphabets ist.

Beachten Sie, dass die Zeichen des Alphabets abhängig sind vom Wert des gesetzten locale. Ist dieser (Voreinstellung) "C", werden die deutschen Umlaute nicht erkannt.

Beispiel 1. ucfirst()-Beispiel:

```
$text = 'markus ist ein kleines schaf.';
$text = ucfirst ($text); // $text ist nun "Markus ist ein kleines
                         // schaf."
```

Siehe auch strtoupper() und strtolower().

ucwords (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Setzt die ersten Buchstaben aller Wörter eines Strings in die entsprechenden Großbuchstaben um

string **ucwords** (string str)

Setzt die ersten Buchstaben aller Wörter innerhalb von *str* in die entsprechenden Großbuchstaben um, sofern es sich um alphabetische Zeichen handelt.

Beachten Sie, dass die Zeichen des Alphabets abhängig sind vom Wert des gesetzten locale. Ist dieser (Voreinstellung) "C", werden die deutschen Umlaute nicht erkannt.

Beispiel 1. ucwords()-Beispiel:

```
$text = "markus ist ein kleines schaf.";
$text = ucwords ($text); // $text ist jetzt: "Markus Ist Ein Kleines
                         // Schaf."
```

Siehe auch strtoupper(), strtolower() und ucfirst().

wordwrap (PHP 4 >= 4.0.2)

Zeilenumbruch eines Strings an einer angegebenen Stelle unter Verwendung eines angegebenen Trennzeichens

string **wordwrap** (string str [, int width [, string break [, int cut]]])

Bricht den String *str* an der mittels *width* angegebenen Position in die nächste Zeile um. Der Umbruch geschieht mit dem *break* Parameter.

Die Funktion **wordwrap()** bricht selbst an der Spalte 75 unter Verwendung des Steuerzeichens '\n' (neue Zeile) um, wenn die optionalen Parameter *width* und / oder *break* nicht angegeben werden.

Steht *cut* auf 1, wird der String immer an der vorgegebenen Länge umgebrochen. Ist ein Wort also länger als die gegebene Länge, wird es mitten drin umgebrochen (siehe zweites Beispiel)

Beispiel 1. wordwrap()-Beispiel:

```
$text = "Der schnelle braune Fuchs springt über den blöden Hund.";
$neuertext = wordwrap( $text, 20 );
echo "$neuertext\n";
```

Dieses Beispiel gibt aus:

```
Der schnelle braune
Fuchs springt über den blöden Hund.
```

Beispiel 2. wordwrap() example

```
$text = "Ein sehr langes Wooooooooooort.";
$neuertext = wordwrap( $text, 8, "\n", 1);
echo "$neuertext\n";
```

Dieses Beispiel gibt aus:

```
Ein sehr
langes W
ooooooo
oort.
```

Siehe auch nl2br().

XCVI. Sybase functions

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get number of affected rows in last query

int **sybase_affected_rows** ([int link_identifier])

Returns: The number of affected rows by the last query.

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use sybase_num_rows().

> **Anmerkung:** This function is only available using the CT library interface to Sybase, and not the DB library.

sybase_close (PHP 3, PHP 4 >= 4.0.0)

close Sybase connection

bool **sybase_close** (int link_identifier)

Returns: TRUE on success, FALSE on error

sybase_close() closes the link to a Sybase database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

sybase_close() will not close persistent links generated by sybase_pconnect().

See also: sybase_connect(), sybase_pconnect().

sybase_connect (PHP 3, PHP 4 >= 4.0.0)

open Sybase server connection

int **sybase_connect** (string servername, string username, string password [, string charset])

Returns: A positive Sybase link identifier on success, or FALSE on error.

sybase_connect() establishes a connection to a Sybase server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **sybase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling sybase_close().

See also sybase_pconnect(), sybase_close().

sybase_data_seek (PHP 3, PHP 4 >= 4.0.0)

move internal row pointer

bool **sybase_data_seek** (int result_identifier, int row_number)

Returns: TRUE on success, FALSE on failure

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result identifier to pointer to the specifyed row number. The next call to sybase_fetch_row() would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4 >= 4.0.0)

fetch row as array

array **sybase_fetch_array** (int result)

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_array() is an extended version of sybase_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **sybase_fetch_array()** is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

For further details, also see sybase_fetch_row().

sybase_fetch_field (PHP 3, PHP 4 >= 4.0.0)

get field information

object **sybase_fetch_field** (int result [, int field_offset])

Returns an object containing field information.

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **sybase_fetch_field()** is retreived.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.

- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric
- type - datatype of the column

See also sybase_field_seek()

sybase_fetch_object (PHP 3, PHP 4 >= 4.0.0)

fetch row as object

int **sybase_fetch_object** (int result)

Returns: An object with properties that correspond to the fetched row, or FALSE if there are no more rows.

sybase_fetch_object() is similar to sybase_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to sybase_fetch_array(), and almost as quick as sybase_fetch_row() (the difference is insignificant).

See also: sybase_fetch_array() and sybase_fetch_row().

sybase_fetch_row (PHP 3, PHP 4 >= 4.0.0)

get row as enumerated array

array **sybase_fetch_row** (int result)

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **sybase_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

See also: sybase_fetch_array(), sybase_fetch_object(), sybase_data_seek(), **sybase_fetch_lengths()**, and sybase_result().

sybase_field_seek (PHP 3, PHP 4 >= 4.0.0)

set field offset

int **sybase_field_seek** (int result, int field_offset)

Seeks to the specified field offset. If the next call to sybase_fetch_field() won't include a field offset, this field would be returned.

See also: sybase_fetch_field().

sybase_free_result (PHP 3, PHP 4 >= 4.0.0)

free result memory

bool **sybase_free_result** (int result)

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

sybase_get_last_message (PHP 3, PHP 4 >= 4.0.0)

Returns the last message from the server

string **sybase_get_last_message** (void)

sybase_get_last_message() returns the last message reported by the server.

sybase_min_client_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum client severity

void **sybase_min_client_severity** (int severity)

sybase_min_client_severity() sets the minimum client severity level.

> **Anmerkung:** This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_server_severity().

sybase_min_error_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum error severity

void **sybase_min_error_severity** (int severity)

sybase_min_error_severity() sets the minimum error severity level.

See also: sybase_min_message_severity().

sybase_min_message_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum message severity

void **sybase_min_message_severity** (int severity)

sybase_min_message_severity() sets the minimum message severity level.

See also: sybase_min_error_severity().

sybase_min_server_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum server severity

void **sybase_min_server_severity** (int severity)

sybase_min_server_severity() sets the minimum server severity level.

> **Anmerkung:** This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_client_severity().

sybase_num_fields (PHP 3, PHP 4 >= 4.0.0)

get number of fields in result

int **sybase_num_fields** (int result)

sybase_num_fields() returns the number of fields in a result set.

See also: **sybase_db_query()**, sybase_query(), sybase_fetch_field(), sybase_num_rows().

sybase_num_rows (PHP 3, PHP 4 >= 4.0.0)

get number of rows in result

int **sybase_num_rows** (int result)

sybase_num_rows() returns the number of rows in a result set.

See also: **sybase_db_query()**, sybase_query() and, sybase_fetch_row().

sybase_pconnect (PHP 3, PHP 4 >= 4.0.0)

open persistent Sybase connection

int **sybase_pconnect** (string servername, string username, string password [, string charset])

Returns: A positive Sybase persistent link identifier on success, or FALSE on error

sybase_pconnect() acts very much like sybase_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (sybase_close() will not close links established by **sybase_pconnect()()**).

This type of links is therefore called 'persistent'.

sybase_query (PHP 3, PHP 4 >= 4.0.0)

send Sybase query

int **sybase_query** (string query, int link_identifier)

Returns: A positive Sybase result identifier on success, or FALSE on error.

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if sybase_connect() was called, and use it.

See also: **sybase_db_query()**, sybase_select_db(), and sybase_connect().

sybase_result (PHP 3, PHP 4 >= 4.0.0)

get result data

string **sybase_result** (int result, int row, mixed field)

Returns: The contents of the cell at the row and offset in the specified Sybase result set.

sybase_result() returns the contents of one cell from a Sybase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: sybase_fetch_row(), sybase_fetch_array(), and sybase_fetch_object().

sybase_select_db (PHP 3, PHP 4 >= 4.0.0)

select Sybase database

bool **sybase_select_db** (string database_name, int link_identifier)

Returns: `TRUE` on success, `FALSE` on error

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if sybase_connect() was called, and use it.

Every subsequent call to sybase_query() will be made on the active database.

See also: sybase_connect(), sybase_pconnect(), and sybase_query()

XCVII. URL Funktionen

base64_decode (PHP 3, PHP 4 >= 4.0.0)

Dekodiert Daten mit MIME base64

string **base64_decode** (string encoded_data)

base64_decode() dekodiert *encoded_data* und bringt die Originaldaten wieder. Die ausgegebenen Dateien können im Binärformat sein.

Siehe auch: base64_encode(), RFC-2045 Sektion 6.8.

base64_encode (PHP 3, PHP 4 >= 4.0.0)

Kodiert Daten MIME base64

string **base64_encode** (string data)

base64_encode() kodiert *data* mit base64. Diese Kodierung ist dafür da, Binärdaten über Technologien, die 8-bit nicht sauber übertragen können, wie z.B. Emails.

Base64-kodierte Daten benötigen ungefähr 33% mehr Speicher als die Originaldaten.

Siehe auch: base64_decode(), chunk_split(), RFC-2045 Sektion 6.8.

parse_url (PHP 3, PHP 4 >= 4.0.0)

Analysiert eine URL und zerlegt diese in ihre Bestandteile.

array **parse_url** (string url)

Diese Funktion erzeugt ein assiziatives Array aus den verschiedenen Teilen einer URL. Diese umfassen "scheme", "host", "port", "user", "pass", "path", "query", und "fragment".

rawurldecode (PHP 3, PHP 4 >= 4.0.0)

Dekodierung von URL-kodierten Strings

string **rawurldecode** (string str)

Gibt einen String zurück, in dem die Abschnitte, die mit einem Prozent-Zeichen (%) beginnen (dem ein 2-stelliger hexadezimal-Wert folgt), ersetzt wurden durch Buchstaben. So wird z.B. die Zeichenkette

```
foo%20bar%40baz
```

umgewandelt in

```
foo bar@baz
```

.

Siehe auch rawurlencode().

rawurlencode (PHP 3, PHP 4 >= 4.0.0)

URL-Kodierung nach RFC 1738

string **rawurlencode** (string str)

Gibt eine Zeichenkette zurück, in der alle Sonderzeichen ausser

`-_.`

durch ein Prozent-Zeichen (%), gefolgt von einer 2-stelligen hexadezimalen Zahl, ersetzt worden sind. Dies ist die Kodierung nach RFC 1738 zum Schutz von Zeichen, die als spezielle URL-Trennzeichen zu interpretieren sind oder die vor einer Umwandlung durch Übertragungs-Medien mit Zeichen-Konvertierung (z.B. einige E-Mail-Systeme) bewahrt werden sollen. Falls sie z.B. ein Passwort in eine FTP-URL einbinden wollen:

Beispiel 1. rawurlencode()-Beispiel 1:

```
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'),
     '@ftp.mein.de/x.txt">';
```

Oder sie wollen Informationen in der Pfadangabe der URL übertragen:

Beispiel 2. rawurlencode()-Beispiel 2:

```
echo '<A HREF="http://x.de/leitung_list_skript/',
    rawurlencode ('Verkauf und Marketing/Hamburg'), '">';
```

Siehe auch rawurldecode().

urldecode (PHP 3, PHP 4 >= 4.0.0)

Dekodiert eine URL-kodierte Zeile

string **urldecode** (string str)

Dekodiert jede %## Kodierung einer Zeile. Die dekodierte Zeile wird zurückgegeben.

Beispiel 1. urldecode() Beispiel

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
    $b = split ('=', $a [$i]);
    echo 'Value for parameter ', htmlspecialchars (urldecode ($b [0])),
```

```
        ' is ', htmlspecialchars (urldecode ($b [1])), "<BR>";
    $i++;
}
```

Siehe auch urlencode(). und rawurlencode().

urlencode (PHP 3, PHP 4 >= 4.0.0)

URL-kodiert eine Zeile

string **urlencode** (string str)

Zeigt eine Zeile an, bei der alle nicht alphanumerischen Zeichen bis auf `-_.` durch das Prozentzeichen (%) ersetzt werden, gefolgt von zwei Hexzeichen und Leerzeichen durch ein Pluszeichen (+) ersetzt werden. Genauso werden auch Daten kodiert, die über ein WWW-Formulart gesendet werden und auf die gleiche Weise mit `application/x-www-form-urlencoded` kodiert werden. Dies unterscheidet sich von der Kodierung in RFC1738 (siehe rawurlencode()) bei dem aus historischen Gründen Leerzeichen wie Pluszeichen (+) kodiert werden. Diese Funktion ist sehr nützlich, wenn eine Zeichenkette kodiert werden soll und die Variablen an die nächste Seite weitergegeben werden sollen:

Beispiel 1. urlencode() Beispiel

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '">';
```

Siehe auch urldecode() und rawurldecode().

XCVIII. Variablen-Funktionen

doubleval (PHP 3, PHP 4 >= 4.0.0)

Konvertiert einen Wert nach double

double **doubleval** (mixed var)

Diese Funktion interpretiert ihr Argument als Fließkommazahl und liefert den Wert als double zurück.

Als *var* können beliebige skalare Ausdrücke, aber keine Arrays oder Objekte übergeben werden.

```
$var = '122.34343The';
$double_value_of_var = doubleval ($var);
print $double_value_of_var; // Ausgabe: 122.34343
```

Siehe auch intval(), strval(), settype() und Typ-Veränderung.

empty (unknown)

Prüft, ob eine Variable einen Wert enthält

int **empty** (mixed var)

Diese Funktion liefert `TRUE`, wenn eine Variable nicht definiert, leer oder gleich 0 ist, ansonsten `FALSE`.

```
$var = 0;
if (empty ($var)) { #evaluates true
    print '$var is either 0 or not at all set';
}
if (!isset ($var)) { // evaluates false
    print 'The $var is not set at all';
}
```

empty() kann nur auf Variablen, nicht aber auf Ausdrücke angewandt werden.

Siehe auch isset() und unset().

gettype (PHP 3, PHP 4 >= 4.0.0)

Liefert den Datentyp einer Variablen

string **gettype** (mixed var)

Diese Funktion liefert den Namen des Typs der Variablen *var* zurück.

Mögliche Ergebnisse sind:

- "integer"
- "double"
- "string"
- "array"
- "object"
- "unknown type"

Siehe auch settype().

intval (PHP 3, PHP 4 >= 4.0.0)

Konvertiert einen Wert nach integer

int **intval** (mixed var [, int base])

Diese Funktion konvertiert den Parameter `var` zu einem integer, wobei `base` als Basis benutzt wird (Default ist `base`=10).

`var` kann ein Ausdruck oder eine einfache Variable sein, nicht jedoch ein Array oder Objekt.

Siehe auch doubleval(), strval(), settype() und Typ-Veränderungen.

is_array (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob Variable ein Array ist

int **is_array** (mixed var)

Liefert `TRUE`, wenn `var` ein Array ist, sonst `FALSE`.

Siehe auch is_double(), is_float(), is_int(), is_integer(), is_real(), is_string(), is_long(), und is_object().

is_bool (PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ bool ist

int **is_bool** (mixed var)

Liefert `TRUE`, wenn `var` vom Typ bool ist.

is_double (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ double ist

int **is_double** (mixed var)

Liefert `TRUE`, wenn *`var`* vom Typ double ist, sonst `FALSE`.

Siehe auch is_array(), is_float(), is_int(), is_integer(), is_real(), is_string(), is_long() und is_object().

is_float (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ float ist

int **is_float** (mixed var)

Diese Funktion ist ein Synonym für is_double().

Siehe auch is_double(), is_real(), is_int(), is_integer(), is_string(), is_object(), is_array() und is_long().

is_int (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ int ist

int **is_int** (mixed var)

Diese Funktion ist ein Synonym für **is_int()**.

Siehe auch is_double(), is_float(), is_integer(), is_string(), is_real(), is_object(), is_array() und is_long().

is_integer (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ integer ist

int **is_integer** (mixed var)

Diese Funktion liefert `TRUE`, wenn *`var`* vom Typ integer ist, sonst `FALSE`.

Siehe auch is_double(), is_float(), is_int(), is_string(), is_real(), is_object(), is_array() und is_long().

is_long (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ long ist

int **is_long** (mixed var)

Diese Funktion ist ein Synonym für is_integer().

Siehe auch is_double(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array() und is_integer().

is_numeric (PHP 4 >= 4.0.0)

Finds whether a variable is a number or a numeric string

int **is_numeric** (mixed var)

Returns `TRUE` if *`var`* is a number or a numeric string, `FALSE` otherwise.

See also is_bool(), is_double(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array(), and is_integer().

is_object (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ object ist

int **is_object** (mixed var)

Liefert `TRUE`, wenn *`var`* ein Objekt enthält, sonst `FALSE`.

Siehe auch is_long(), is_int(), is_integer(), is_float(), is_double(), is_real(), is_string() und is_array().

is_real (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob Variable vom Typ real ist

int **is_real** (mixed var)

Diese Funktion ist ein Synonym für is_double().

Siehe auch is_long(), is_int(), is_integer(), is_float(), is_double(), is_object(), is_string() und is_array().

is_resource (PHP 4 >= 4.0.0)

Prüft, ob eine Variable vom Typ resource ist

int **is_resource** (mixed var)

is_resource() liefert `TRUE`, wenn es sich bei der Variable *`var`* um eine PHP-Resource wie z.B. ein Datei- oder Result-Handle handelt, ansonsten `FALSE`. PHP-Resorucen werden von PHP-internen Funktione erzeugt und freigegeben. Unterläst der Benutzer die Freigabe in seinem Code, so kann diese durch den speziellen Datentyp resource dennoch angestossen werden, sobald die Resource-Variable nicht mehr referenziert wird.

is_string (PHP 3, PHP 4 >= 4.0.0)

Prüft, ob Variable vom Typ string ist

int **is_string** (mixed var)

Liefert `TRUE`, wenn *`var`* vom Typ string ist, sonst `FALSE`.

Siehe auch is_long(), is_int(), is_integer(), is_float(), is_double(), is_real(), is_object() und is_array().

isset (unknown)

Prüft die Existenz einer Variablen

int **isset** (mixed var)

Diese Funktion liefert `TRUE`, wenn die Variable oder das Array-Element *`var`* existiert, sonst `FALSE`.

Wird eine Variable oder ein Array-Element mit unset() freigegeben, so liefert **isset()** anschließend ebenfalls `FALSE`.

```
echo isset ($a); // false
$a = "test";
echo isset ($a); // true
unset ($a);
echo isset ($a); // false
```

Siehe auch empty() und unset().

print_r (PHP 4 >= 4.0.0)

Gibt Variablen-Informationen in lesbarer Form aus

void **print_r** (mixed expression)

Diese Funktion zeigt Informationen über eine Variable oder einen Ausdruck in lesbarer Form an. Einfache Werte werden direkt ausgegeben, Arrays und Objekte werden dagegen als eingerückte Schlüssel / Wert-Paare ausgegeben.

print_r() ähnelt var_dump(), gibt aber Typinformationen nur bei Arrays und Objekten aus.

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
print_r ($a);
?>
```

Warnung

Diese Funktion ist nicht in der Lage, zyklische Referenzen aufzulösen, d.h. die Ausgabe eines Arrays oder Objektes, das sich selbst enthält, wie z.B. `print_r($GLOBALS)`, wird niemals beendet.

serialize (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Erzeugt ein speicherbares Datenformat

string **serialize** (mixed value)

serialize() gibt eine Zeichenkette (String) zurück, die eine Byte-Stream entsprechende Wiedergabe von *value* enthält und beliebig abgespeichert werden kann.

Diese Funktion dient der Speicherung oder Übergabe von PHP-Werten, ohne dass diese ihren Wert oder ihre Struktur verlieren.

Um eine serialisierte Zeichenkette wieder als PHP-Wert verfügbar zu machen steht die unserialize()-Funktion zur Verfügung. **serialize()** kann mit den Typen integer, double, string, array (mehrdimensional) und object umgehen. Beim Objekt werden die Eigenschaften serialisiert, die Methoden gehen aber verloren.

Beispiel 1. serialize()-Beispiel:

```
// $session_data enthält ein mehrdimensionales Array mit
// Session-Informationen des aktuellen Benutzers. Wir benutzen
// serialize() um diese Infos am Ende der Anfrage in eine
// Datenbank zu speichern.

$conn = odbc_connect ("webdb", "php", "moorhuhn");
$stmt = odbc_prepare ($conn,
                      "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata)) {
    $stmt = odbc_prepare ($conn,
                          "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
        /* Etwas lief schief. Fluche, heule und stöhne. */
    }
}
```

settype (PHP 3, PHP 4 >= 4.0.0)

Legt den Typ einer Variablen fest

int **settype** (string var, string type)

Der Typ der Variablen `var` wird festgelegt als `type`.

Mögliche Werte für `type` sind:

- "integer"
- "double"
- "string"
- "array"
- "object"

Liefert `TRUE` bei Erfolg, sonst `FALSE`.

Siehe auch gettype().

strval (PHP 3, PHP 4 >= 4.0.0)

Konvertierung zum String

string **strval** (mixed var)

Diese Funktion liefert den übergebenen Ausdruck `var` als String-Repräsentation zurück.

`var` darf von jedem einfache Datentyp, aber kein Array oder Objekt sein.

Siehe auch doubleval(), intval(), settype() und Typ-Veränderung.

unserialize (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Erzeugt aus einem gespeicherten Datenformat einen Wert in PHP

mixed **unserialize** (string str)

unserialize() nimmt eine einzelne Variable aus einer serialisierten Datenmenge (siehe unter serialize()) und wandelt sie in einen PHP-Wert um. Dieser Wert wird als Funktionsergebnis zurück gegeben. Er kann vom Typ integer, double, string, array oder object sein. Wurde ein Objekt serialisiert, können dessen Methoden nicht zurück gegeben werden.

Beispiel 1. unserialize()-Beispiel:

```
// Als Beispiel benutzen wir unserialize() um Daten aus einer
// Datenbank nach $session_data zu laden. Das Beispiel vervollständigt
// das unter serialize() beschriebene.

$conn = odbc_connect ("webdb", "php", "moorhuhn");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // wenn execute oder fetch fehlschlägt, initialisiere ein leeres Array
    $session_data = array();
```

```
} else {
    // die serialisierten Daten sollten nun in $tmp[0] stehen
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
        // etwas ging schief, also wieder ein leeres Array initialisieren
        $session_data = array();
    }
}
```

unset (unknown)

Löschen einer Variablen

int **unset** (mixed var)

unset() gibt eine Variable oder ein Array-Element vollständig frei.

Beispiel 1. unset() example

```
unset ($foo);
unset ($bar['quux']);
```

Siehe auch isset() und empty().

var_dump (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Gibt alle Informationen zu einer Variablen aus

void **var_dump** (mixed expression)

Diese Fuktion liefert Informationen über Typ und Wert des Parameter *`expression`* zurück. Arrays und Objekte werden rekursiv durchlaufen und mit entsprechender Einrückung dargestellt.

Siehe auch print_r().

```
<pre>
<?php
    $a = array (1, 2, array ("a", "b", "c"));
    var_dump ($a);
?>
</pre>
```

XCIX. vpopmail functions

vpopmail_add_domain (PHP 4 >= 4.0.5)

Add a new virtual domain

bool **vpopmail_add_domain** (string domain, string dir, int uid, int gid)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_del_domain (PHP 4 >= 4.0.5)

Delete a virtual domain

bool **vpopmail_del_domain** (string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_add_alias_domain (PHP 4 >= 4.0.5)

Add an alias for a virtual domain

bool **vpopmail_add_alias_domain** (string domain, string aliasdomain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_add_domain_ex (PHP 4 >= 4.0.5)

Add a new virtual domain

bool **vpopmail_add_domain_ex** (string domain, string passwd [, string quota [, string bounce [, bool apop]]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_del_domain_ex (PHP 4 >= 4.0.5)

Delete a virtual domain

bool **vpopmail_del_domain_ex** (string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_add_alias_domain_ex (PHP 4 >= 4.0.5)

Add alias to an existing virtual domain

bool **vpopmail_add_alias_domain_ex** (string olddomain, string newdomain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_add_user (PHP 4 >= 4.0.5)

Add a new user to the specified virtual domain

bool **vpopmail_add_user** (string user, string domain, string password [, string gecos [, bool apop]])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_del_user (PHP 4 >= 4.0.5)

Delete a user from a virtual domain

bool **vpopmail_del_user** (string user, string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_passwd (PHP 4 >= 4.0.5)

Change a virtual user's password

bool **vpopmail_passwd** (string user, string domain, string password)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_set_user_quota (PHP 4 >= 4.0.5)

Sets a virtual user's quota

bool **vpopmail_set_user_quota** (string user, string domain, string quota)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_auth_user (PHP 4 >= 4.0.5)

Attempt to validate a username/domain/password. Returns true/false

bool **vpopmail_auth_user** (string user, string domain, string password [, string apop])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_alias_add (PHP 4 >= 4.1.0)

insert a virtual alias

bool **vpopmail_alias_add** (string user, string domain, string alias)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_alias_del (PHP 4 >= 4.1.0)

deletes all virtual aliases of a user

bool **vpopmail_alias_del** (string user, string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_alias_del_domain (PHP 4 >= 4.1.0)

deletes all virtual aliases of a domain

bool **vpopmail_alias_del_domain** (string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_alias_get (PHP 4 >= 4.1.0)

get all lines of an alias for a domain

array **vpopmail_alias_get** (string alias, string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_alias_get_all (PHP 4 >= 4.1.0)

get all lines of an alias for a domain

array **vpopmail_alias_get_all** (string domain)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

vpopmail_error (PHP 4 >= 4.0.5)

Get text message for last vpopmail error. Returns string

string **vpopmail_error** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

C. W32api functions

w32api_set_call_method (PHP 4 CVS only)

Sets the calling method used

void **w32api_set_call_method** (int method)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

w32api_register_function (PHP 4 CVS only)

Registers function function_name from library with PHP

bool **w32api_register_function** (string libary, string function_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

w32api_invoke_function (unknown)

....) Invokes function funcname with the arguments passed after the function name

mixed **w32api_invoke_function** (string funcname)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

w32api_deftype (PHP 4 CVS only)

...) Defines a type for use with other w32api_functions.

int **w32api_deftype** (string typename, string member1_type, string member1_name)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

w32api_init_dtype (PHP 4 CVS only)

; Creates an instance to the data type typename and fills it with the values val1, val2, the function

resource **w32api_init_dtype** (string typename, mixed val1, mixed val2)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

CI. WDDX Funktionen

Diese Funktionen arbeiten mit WDDX (http://www.openwddx.org/) zusammen.

Es ist zu beachten, dass alle Funktionen die Variablen serialisieren immer das erste Element eines Arrays benutzen, um festzustellen ob das Array in ein Array oder ein 'Structure' serialisiert wird. Wenn das erste Element einen String als Schlüssel hat, wird das Array in eine 'Structure' serialisiert, andernfalls in ein Array.

Beispiel 1. Einen einzelnen String serialisieren

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

dieses Beispiel erzeugt:

```
<wddxPacket version='1.0'><header comment='PHP packet'/><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Beispiel 2. Die Verwendung von inkrementierenden Paketen

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* So tun, als ob $cities aus einer Datenbank kommt */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

Diese Beispiel wird erzeugen:

```
<wddxPacket version='1.0'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

wddx_serialize_value (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Serialisiert einen einzelnen Wert in ein WDDX Paket

string **wddx_serialize_value** (mixed wert [, string kommentar])

wddx_serialize_value() wird verwendet um einen einzelnen Wert in ein WDDX Paket zu serialisieren. Übergeben wird der Wert in *wert* und ein optionaler Parameter *kommentar*, der im Header des WDDX Pakets erscheinen wird. Zurückgegeben wird das entsprechende WDDX Paket.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Serialisiert Variablen in WDDX Pakete

string **wddx_serialize_vars** (mixed var_name [, mixed ...])

wddx_serialize_vars() wird verwendet um ein WDDX Paket mit einer 'Structure' zu erzeugen, welches eine serialsierte Repräsentation der übergebenen Variablen enthält.

wddx_serialize_vars() akzeptiert eine variable Zahl von Argumenten. Diese können entweder einzelne Namen von Variablen, oder Arrays mit Variablennamen bzw. weiteren Arrays sein.

Beispiel 1. wddx_serialize_vars example

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

Das obige Script erzeugt:

```
<wddxPacket version='1.0'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

wddx_packet_start (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Beginnt ein neues WDDX Paket mit einer 'Structure'

int **wddx_packet_start** ([string kommentar])

wddx_packet_start() wird verwendet um ein neues WDDX Paket zu erzeugen, zu welchem Variablen hinzugefügt werden können. Ein optionaler Parameter `kommentar` vom Typ String kann angegeben werden. Zurückgegeben wird eine Paket ID, die in den folgenden Funktionen zur Referenzierung benutzt wird. Desweiteren wird automatisch eine 'Structure' zur Aufnahme der Variablen angelegt.

wddx_packet_end (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Schliesst das WDDX Paket mit der angegebenen ID

string **wddx_packet_end** (int packet_id)

wddx_packet_end() schliesst das WDDX Paket, dass mit `packet_id` identifiziert wird und gibt einen String, der das erzeugte WDDX Paket enthält zurück.

wddx_add_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Fügt dem WDDX Paket mit der übergebenen ID Werte hinzu

wddx_add_vars (int packet_id, mixed name_var [, mixed ...])

wddx_add_vars() wird verwendet um die übergebenen Werte zu serialisieren und diese an das durch den Parameter `packet_id` referenzierte Paket anzuhängen. Die Werteübergabe funktioniert genauso wie bei wddx_serialize_vars().

wddx_deserialize (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Deserialisiert ein WDDX Paket

mixed **wddx_deserialize** (string packet)

wddx_deserialize() übernimmt einen `packet` String and deserialisiert diesen. Es wird entweder ein String, eine Zahl, oder ein Array zurückgegeben. 'Structures' werden in assoziative Arrays deserialisiert.

CII. XML parser functions

Introduction

About XML

XML (eXtensible Markup Language) is a data format for structured document interchange on the Web. It is a standard defined by The World Wide Web consortium (W3C). Information about XML and related technologies can be found at http://www.w3.org/XML/.

Installation

This extension uses expat, which can be found at http://www.jclark.com/xml/. The Makefile that comes with expat does not build a library by default, you can use this make rule for that:

```
libexpat.a: $(OBJS)
    ar -rc $@ $(OBJS)
    ranlib $@
```

A source RPM package of expat can be found at http://sourceforge.net/projects/expat/.

Note that if you are using Apache-1.3.7 or later, you already have the required expat library. Simply configure PHP using `--with-xml` (without any additional path) and it will automatically use the expat library built into Apache.

On UNIX, run **configure** with the `--with-xml` option. The expat library should be installed somewhere your compiler can find it. If you compile PHP as a module for Apache 1.3.9 or later, PHP will automatically use the bundled expat library from Apache. You may need to set CPPFLAGS and LDFLAGS in your environment before running configure if you have installed expat somewhere exotic.

Build PHP. *Tada!* That should be it.

About This Extension

This PHP extension implements support for James Clark's expat in PHP. This toolkit lets you parse, but not validate, XML documents. It supports three source character encodings also provided by PHP: `US-ASCII`, `ISO-8859-1` and `UTF-8`. `UTF-16` is not supported.

This extension lets you create XML parsers and then define *handlers* for different XML events. Each XML parser also has a few parameters you can adjust.

The XML event handlers defined are:

Tabelle 1. Supported XML handlers

PHP function to set handler	Event description

PHP function to set handler	Event description
xml_set_element_handler()	Element events are issued whenever the XML parser encounters start or end tags. There are separate handlers for start tags and end tags.
xml_set_character_data_handler()	Character data is roughly all the non-markup contents of XML documents, including whitespace between tags. Note that the XML parser does not add or remove any whitespace, it is up to the application (you) to decide whether whitespace is significant.
xml_set_processing_instruction_handler()	PHP programmers should be familiar with processing instructions (PIs) already. <?php ?> is a processing instruction, where *php* is called the "PI target". The handling of these are application-specific, except that all PI targets starting with "XML" are reserved.
xml_set_default_handler()	What goes not to another handler goes to the default handler. You will get things like the XML and document type declarations in the default handler.
xml_set_unparsed_entity_decl_handler()	This handler will be called for declaration of an unparsed (NDATA) entity.
xml_set_notation_decl_handler()	This handler is called for declaration of a notation.
xml_set_external_entity_ref_handler()	This handler is called when the XML parser finds a reference to an external parsed general entity. This can be a reference to a file or URL, for example. See the external entity example for a demonstration.

Case Folding

The element handler functions may get their element names *case-folded*. Case-folding is defined by the XML standard as "a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents". In other words, when it comes to XML, case-folding simply means uppercasing.

By default, all the element names that are passed to the handler functions are case-folded. This behaviour can be queried and controlled per XML parser with the xml_parser_get_option() and xml_parser_set_option() functions, respectively.

Error Codes

The following constants are defined for XML error codes (as returned by xml_parse()):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY

XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Character Encoding

PHP's XML extension supports the Unicode (http://www.unicode.org/) character set through different *character encoding*s. There are two types of character encodings, *source encoding* and *target encoding*. PHP's internal representation of the document is always encoded with `UTF-8`.

Source encoding is done when an XML document is parsed. Upon creating an XML parser, a source encoding can be specified (this encoding can not be changed later in the XML parser's lifetime). The supported source encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`. The former two are single-byte encodings, which means that each character is represented by a single byte. `UTF-8` can encode characters composed by a variable number of bits (up to 21) in one to four bytes. The default source encoding used by PHP is `ISO-8859-1`.

Target encoding is done when PHP passes data to XML handler functions. When an XML parser is created, the target encoding is set to the same as the source encoding, but this may be changed at any point. The target encoding will affect character data as well as tag names and processing instruction targets.

If the XML parser encounters characters outside the range that its source encoding is capable of representing, it will return an error.

If PHP encounters characters in the parsed XML document that can not be represented in the chosen target encoding, the problem characters will be "demoted". Currently, this means that such characters

are replaced by a question mark.

Some Examples

Here are some example PHP scripts parsing XML documents.

XML Element Structure Example

This first example displays the stucture of the start elements in a document with indentation.

Beispiel 1. Show XML Element Structure

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML Tag Mapping Example

Beispiel 2. Map XML to HTML

This example maps tags in an XML document directly to HTML tags. Elements not found in the "map array" are ignored. Of course, this example will only work with a specific XML document type.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML External Entity Example

This example highlights XML code. It illustrates how to use an external entity reference handler to include and parse other documents, as well as how PIs can be processed, and a way of determining "trust" for PIs containing code.

XML documents that can be used for this example are found below the example (`xmltest.xml` and `xmltest2.xml`.)

Beispiel 3. External Entity Example

```
$file = "xmltest.xml";

function trustedFile($file) {
    // only trust local files owned by ourselves
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // If the parsed document is "trusted", we say it is safe
            // to execute PHP code inside it.  If not, display the code
            // instead.
            if (trustedFile($parser_file[$parser])) {
```

```
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}


function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
```

```
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);

?>
```

Beispiel 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!-- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
```

```
 </sect1>
</chapter>
```

This file is included from `xmltest.xml`:

Beispiel 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

xml_parser_create (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

create an XML parser

int **xml_parser_create** ([string encoding])

encoding (optional)

Which character encoding the parser should use. The following character encodings are supported:
`ISO-8859-1` (default)
`US-ASCII`
`UTF-8`

This function creates an XML parser and returns a handle for use by other XML functions. Returns `FALSE` on failure.

xml_set_object (PHP 4 >= 4.0.0)

Use XML Parser within an object

void **xml_set_object** (int parser, object &object)

This function allows to use *parser* inside *object*. All callback functions could be set with xml_set_element_handler() etc and assumed to be methods of *object*.

```
<?php
class xml  {
var $parser;

function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,&$this);
    xml_set_element_handler($this->parser,"tag_open","tag_close");
    xml_set_character_data_handler($this->parser,"cdata");
}

function parse($data) {
    xml_parse($this->parser,$data);
}

function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}

function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}

function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}
```

```
} // end of class xml

$xml_parser = new xml();
$xml_parser->parse("<A ID=\"hallo\">PHP</A>");
?>
```

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

set up start and end element handlers

int **xml_set_element_handler** (int parser, string startElementHandler, string endElementHandler)

Sets the element handler functions for the XML parser *parser*. *startElementHandler* and *endElementHandler* are strings containing the names of functions that must exist when xml_parse() is called for *parser*.

The function named by *startElementHandler* must accept three parameters:
startElementHandler (int parser, string name, array attribs)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

attribs

The third parameter, *attribs*, contains an associative array with the element's attributes (if any). The keys of this array are the attribute names, the values are the attribute values. Attribute names are case-folded on the same criteria as element names. Attribute values are *not* case-folded.

The original order of the attributes can be retrieved by walking through *attribs* the normal way, using each(). The first key in the array was the first attribute, and so on.

The function named by *endElementHandler* must accept two parameters:
endElementHandler (int parser, string name)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handlers are set up, FALSE if *parser* is not a parser.

> **Anmerkung:** Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

set up character data handler

int **xml_set_character_data_handler** (int parser, string handler)

Sets the character data handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

The function named by *handler* must accept two parameters: ***handler*** (int parser, string data)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data as a string.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

> **Anmerkung:** Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set up processing instruction (PI) handler

int **xml_set_processing_instruction_handler** (int parser, string handler)

Sets the processing instruction (PI) handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

A processing instruction has the following format:

```
<?
        target
        data?>
```

You can put PHP code into such a tag, but be aware of one limitation: in an XML PI, the PI end tag (?>) can not be quoted, so this character sequence should not appear in the PHP code you embed with PIs in XML documents. If it does, the rest of the PHP code, as well as the "real" PI end tag, will be treated as character data.

The function named by *handler* must accept three parameters: ***handler*** (int parser, string target, string data)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

target

The second parameter, *target*, contains the PI target.

data

The third parameter, *data*, contains the PI data.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

set up default handler

int **xml_set_default_handler** (int parser, string handler)

Sets the default handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

The function named by *handler* must accept two parameters: ***handler*** (int parser, string data)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data. This may be the XML declaration, document type declaration, entities or other data for which no other handler exists.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set up unparsed entity declaration handler

int **xml_set_unparsed_entity_decl_handler** (int parser, string handler)

Sets the unparsed entity declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

This handler will be called if the XML parser encounters an external entity declaration with an NDATA declaration, like the following:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

See section 4.2.2 of the XML 1.0 spec (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) for the definition of notation declared external entities.

The function named by *handler* must accept six parameters: ***handler*** (int parser, string entityName, string base, string systemId, string publicId, string notationName)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

entityName

The name of the entity that is about to be defined.

base

This is the base for resolving the system identifier (*systemId*) of the external entity. Currently this parameter will always be set to an empty string.

systemId

System identifier for the external entity.

publicId

Public identifier for the external entity.

notationName

Name of the notation of this entity (see xml_set_notation_decl_handler()).

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

set up notation declaration handler

int **xml_set_notation_decl_handler** (int parser, string handler)

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

A notation declaration is part of the document's DTD and has the following format:

```
<!NOTATION
     name {systemId |
     publicId}>
```

See section 4.7 of the XML 1.0 spec (http://www.w3.org/TR/1998/REC-xml-19980210#Notations) for the definition of notation declarations.

The function named by *handler* must accept five parameters: ***handler*** (int parser, string notationName, string base, string systemId, string publicId)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

notationName

This is the notation's *name*, as per the notation format described above.

base

This is the base for resolving the system identifier (*systemId*) of the notation declaration. Currently this parameter will always be set to an empty string.

systemId

System identifier of the external notation declaration.

publicId

Public identifier of the external notation declaration.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

set up external entity reference handler

int **xml_set_external_entity_ref_handler** (int parser, string handler)

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when xml_parse() is called for *parser*.

The function named by *handler* must accept five parameters, and should return an integer value. If the value returned from the handler is FALSE (which it will be if no value is returned), the XML parser will stop parsing and xml_get_error_code() will return XML_ERROR_EXTERNAL_ENTITY_HANDLING. int ***handler*** (int parser, string openEntityNames, string base, string systemId, string publicId)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

openEntityNames

The second parameter, *openEntityNames*, is a space-separated list of the names of the entities that are open for the parse of this entity (including the name of the referenced entity).

base

This is the base for resolving the system identifier (*systemid*) of the external entity. Currently this parameter will always be set to an empty string.

systemId

The fourth parameter, *systemId*, is the system identifier as specified in the entity declaration.

publicId

The fifth parameter, *publicId*, is the public identifier as specified in the entity declaration, or an empty string if none was specified; the whitespace in the public identifier will have been normalized as required by the XML spec.

If a handler function is set to an empty string, or FALSE, the handler in question is disabled.

TRUE is returned if the handler is set up, FALSE if *parser* is not a parser.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_parse (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

start parsing an XML document

int **xml_parse** (int parser, string data [, int isFinal])

parser

A reference to the XML parser to use.

data

Chunk of data to parse. A document may be parsed piece-wise by calling **xml_parse()** several times with new data, as long as the *isFinal* parameter is set and TRUE when the last data is parsed.

isFinal (optional)

If set and TRUE, *data* is the last piece of data sent in this parse.

When the XML document is parsed, the handlers for the configured events are called as many times as necessary, after which this function returns TRUE or FALSE.

TRUE is returned if the parse was successful, FALSE if it was not successful, or if *parser* does not refer to a valid parser. For unsuccessful parses, error information can be retrieved with xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number() and xml_get_current_byte_index().

xml_get_error_code (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get XML parser error code

int **xml_get_error_code** (int parser)

parser

A reference to the XML parser to get error code from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns one of the error codes listed in the error codes section.

xml_error_string (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get XML parser error string

string **xml_error_string** (int code)

code

An error code from xml_get_error_code().

Returns a string with a textual description of the error code *code*, or FALSE if no description was found.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get current line number for an XML parser

int **xml_get_current_line_number** (int parser)

parser

A reference to the XML parser to get line number from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns which line the parser is currently at in its data buffer.

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get current column number for an XML parser

int **xml_get_current_column_number** (int parser)

parser

A reference to the XML parser to get column number from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns which column on the current line (as given by xml_get_current_line_number()) the parser is currently at.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get current byte index for an XML parser

int **xml_get_current_byte_index** (int parser)

parser

A reference to the XML parser to get byte index from.

This function returns FALSE if *parser* does not refer to a valid parser, or else it returns which byte index the parser is currently at in its data buffer (starting at 0).

xml_parse_into_struct (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Parse XML data into an array structure

int **xml_parse_into_struct** (int parser, string data, array &values, array &index)

This function parses an XML file into 2 parallel array structures, one (*index*) containing pointers to the location of the appropriate values in the *values* array. These last two parameters must be passed by reference.

Below is an example that illustrates the internal structure of the arrays being generated by the function. We use a simple `note` tag embeded inside a `para` tag, and then we parse this an print out the structures generated:

```
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Index array\n";
print_r($index);
echo "\nVals array\n";
print_r($vals);
```

When we run that code, the output will be:

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
```

```
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

Event-driven parsing (based on the expat library) can get complicated when you have an XML document that is complex. This function does not produce a DOM style object, but it generates structures amenable of being transversed in a tree fashion. Thus, we can create objects representing the data in the XML file easily. Let's consider the following XML file representing a small database of aminoacids information:

Beispiel 1. moldb.xml - small database of molecular information

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>

</moldb>
```

And some code to parse the document and generate the appropriate objects:

Beispiel 2. parsemoldb.php - parses moldb.xml into and array of molecular objects

```
<?php

class AminoAcid {
    var $name;  // aa name
    var $symbol;    // three letter symbol
    var $code;  // one letter code
```

```
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa) {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
    }
}

function readDatabase($filename) {
    // read the xml database of aminoacids
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,$values,$tags);
    xml_parser_free($parser);

    // loop through the structures
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // each contiguous pair of array entries are the
            // lower and upper range for each molecule definition
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}

function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
print_r($db);

?>
```

After executing `parsemoldb.php`, the variable `$db` contains an array of `AminoAcid` objects, and the output of the script confirms that:

```
** Database of AminoAcid objects:
Array
(
```

```
        [0] => aminoacid Object
            (
                [name] => Alanine
                [symbol] => ala
                [code] => A
                [type] => hydrophobic
            )

        [1] => aminoacid Object
            (
                [name] => Lysine
                [symbol] => lys
                [code] => K
                [type] => charged
            )

    )
```

xml_parser_free (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Free an XML parser

string **xml_parser_free** (int parser)

parser

A reference to the XML parser to free.

This function returns `FALSE` if *`parser`* does not refer to a valid parser, or else it frees the parser and returns `TRUE`.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

set options in an XML parser

int **xml_parser_set_option** (int parser, int option, mixed value)

parser

A reference to the XML parser to set an option in.

option

Which option to set. See below.

value

The option's new value.

This function returns FALSE if *parser* does not refer to a valid parser, or if the option could not be set. Else the option is set and TRUE is returned.

The following options are available:

Tabelle 1. XML parser options

Option constant	Data type	Description
XML_OPTION_CASE_FOLDING	integer	Controls whether case-folding is enabled for this XML parser. Enabled by default.
XML_OPTION_TARGET_ENCODING	string	Sets which target encoding to use in this XML parser. By default, it is set to the same as the source encoding used by xml_parser_create(). Supported target encodings are ISO-8859-1, US-ASCII and UTF-8.

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get options from an XML parser

mixed **xml_parser_get_option** (int parser, int option)

parser

A reference to the XML parser to get an option from.

option

Which option to fetch. See xml_parser_set_option() for a list of options.

This function returns FALSE if *parser* does not refer to a valid parser, or if the option could not be set. Else the option's value is returned.

See xml_parser_set_option() for the list of options.

utf8_decode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1.

string **utf8_decode** (string data)

This function decodes *data*, assumed to be `UTF-8` encoded, to `ISO-8859-1`.

See utf8_encode() for an explaination of UTF-8 encoding.

utf8_encode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

encodes an ISO-8859-1 string to UTF-8

string **utf8_encode** (string data)

This function encodes the string *data* to `UTF-8`, and returns the encoded version. `UTF-8` is a standard mechanism used by Unicodefor encoding *wide character* values into a byte stream. `UTF-8` is transparent to plain ASCII characters, is self-synchronized (meaning it is possible for a program to figure out where in the bytestream characters start) and can be used with normal string comparison functions for sorting and such. PHP encodes `UTF-8` characters in up to four bytes, like this:

Tabelle 1. UTF-8 encoding

bytes	bits	representation
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Each *b* represents a bit that can be used to store character data.

xml_parser_create_ns (PHP 4 >= 4.0.5)

Create an XML parser

int **xml_parser_create_ns** ([string encoding [, string sep]])

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xml_set_end_namespace_decl_handler (PHP 4 >= 4.0.5)

Set up character data handler

int **xml_set_end_namespace_decl_handler** (int pind, string hdl)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

xml_set_start_namespace_decl_handler (PHP 4 >= 4.0.5)

Set up character data handler

int **xml_set_start_namespace_decl_handler** (int pind, string hdl)

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

Anmerkung: Anstelle eines Funktionsnamens kann aber auch ein Array mit einer Objekt-Referenz und dem Namen einer Methode angegeben werden.

CIII. XMLRPC functions

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

xmlrpc_encode_request (PHP 4 >= 4.1.0)

Generates XML for a method request

string **xmlrpc_encode_request** (string method, mixed params)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_encode (PHP 4 >= 4.1.0)

Generates XML for a PHP value

string **xmlrpc_encode** (mixed value)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_decode_request (PHP 4 >= 4.1.0)

Decodes XML into native PHP types

array **xmlrpc_decode_request** (string xml, string method [, string encoding])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_decode (PHP 4 >= 4.1.0)

Decodes XML into native PHP types

array **xmlrpc_decode** (string xml [, string encoding])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_server_create (PHP 4 >= 4.1.0)

Creates an xmlrpc server

resource **xmlrpc_server_create** (void)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_server_destroy (PHP 4 >= 4.1.0)

Destroys server resources

void **xmlrpc_server_destroy** (resource server)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_server_register_method (PHP 4 >= 4.1.0)

Register a PHP function to resource method matching method_name

bool **xmlrpc_server_register_method** (resource server, string method_name, string function)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_server_register_introspection_callback (PHP 4 >= 4.1.0)

Register a PHP function to generate documentation

bool **xmlrpc_server_register_introspection_callback** (resource server, string function)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_server_call_method (PHP 4 >= 4.1.0)

Parses XML requests and call methods

mixed **xmlrpc_server_call_method** (resource server, string xml, mixed user_data [, array output_options])

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_server_add_introspection_data (PHP 4 >= 4.1.0)

Adds introspection documentation

int **xmlrpc_server_add_introspection_data** (resource server, array desc)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_parse_method_descriptions (PHP 4 >= 4.1.0)

Decodes XML into a list of method descriptions

array **xmlrpc_parse_method_descriptions** (string xml)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_set_type (PHP 4 >= 4.1.0)

Sets xmlrpc type, base64 or datetime, for a PHP string value

bool **xmlrpc_set_type** (string value, string type)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

xmlrpc_get_type (PHP 4 >= 4.1.0)

Gets xmlrpc type for a PHP value. Especially useful for base64 and datetime strings

string **xmlrpc_get_type** (mixed value)

Warnung

Diese Funktion ist *EXPERIMENTELL*. Das Verhalten, der Funktionsname und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Funktion auf eigenes Risiko.

Warnung

Diese Funktion ist bis jetzt nicht dokumentiert. Es steht nur die Liste der Argumente zur Verfügung.

CIV. XSLT Funktionen

Warnung

Diese Erweiterung ist *EXPERIMENTELL*. Das Verhalten dieser Erweiterung, einschließlich der Funktionsnamen, und alles Andere was hier dokumentiert ist, kann sich in zukünftigen PHP-Versionen ohne Ankündigung ändern. Seien Sie gewarnt und verwenden Sie diese Erweiterung auf eigenes Risiko.

Einführung

Über XSLT und Sablotron

XSLT (Extensible Stylesheet Language (XSL) Transformations) ist eine Sprache zur Transformation von XML Dokumenten in andere XML Dokumente. XLST ist als Standard vom World Wide Web (W3C) Konsortium definiert worden. Informationen zu XSLT und diesbezüglichen Technologien können unter http://www.w3.org/TR/xslt gefunden werden.

Installation

Die Erweiterung verwendet Sabloton und expat, wobei beide auf http://www.gingerall.com/ zu finden sind. Es werden sowohl Binaries, wie auch die Quelldateien angeboten.

Die Erweiterung ist unter UNIX mit dem Befehl **configure** und den Optionen `--with-sablot` und `--enable-sablot-errors-descriptive` einzubinden. Die Sablotron Library sollte so auf dem Server installiert sein, dass sie der Compiler finden kann.

Über diese Erweiterung

Die PHP Erweiterung bindet Sablotron von Ginger Alliance in PHP ein. Dieses Toolkit erlaubt die Transformation von XML Dokumenten in andere XML Dokumente, HTML und andere Formate. Die Erweiterung bietet standardiesierte und portable Template Mechanismen zur Trennung von Daten und Design bei Websites.

xslt_closelog (4.0.3 - 4.0.6 only)

Löschen der Log-Datei zu einer gegebenen Sablotron Instanz

bool **xslt_closelog** (resource xh)

xh

Eine Referenz zum XSLT Parser.

Die Funktion gibt `FALSE` zurück, wenn *`parser`* nicht auf einen gültigen parser verweist, oder das Schliessen der Log-Datei fehlgeschlagen ist. Ansonsten wird `TRUE` zurückgegeben.

xslt_create (PHP 4 >= 4.0.3)

Einen neuen XSL processor erzeugen.

resource **xslt_create** (void)

Die Funktion gibt einen handle eines neuen XSL Prozessors zurück. Dieser handle wird für alle folgenden Aufrufe von XSL Funktionen zu verwenden sein.

xslt_errno (PHP 4 >= 4.0.3)

Gibt die aktuelle Fehlernummer zurück

int **xslt_errno** ([int xh])

Gibt die aktuelle Fehlernummer zu einem gegebenen XSL Prozessor zurück. Ist kein handle gegeben, so wird die Fehlernummer des zuletzt irgendwo aufgetretenen Fehlers zurückgegeben.

xslt_error (PHP 4 >= 4.0.3)

Gibt den aktuellen Fehlerstring zurück.

mixed **xslt_error** ([int xh])

Gibt den aktuellen Fehlerstring zu einem gegebenen XSL Prozessor zurück. Ist kein handle gegeben, so wird die Fehlerstring des zuletzt irgendwo aufgetretenen Fehlers zurückgegeben.

xslt_fetch_result (4.0.3 - 4.0.6 only)

Ruft einen (benannte) Resultatspeich ab

string **xslt_fetch_result** (int xh [, string result_name])

Ruft einen (benannte) Resultatspeicher des XSLT Prozessors ab, welcher durch den handle identifiziert worden ist. Ist der Speicher nicht ausdrücklich benannt worden, so wird der abgerufene Specherinhalt mit "/_result" benannt.

xslt_free (PHP 4 >= 4.0.3)

Freigabe eines XSLT Prozessors

void **xslt_free** (resource xh)

Der mit handle bezeichnete XSLT Prozessor wird freigegeben.

xslt_openlog (4.0.3 - 4.0.6 only)

Benennen einer Log-Datei für XSLT Prozessor-Meldungen

bool **xslt_openlog** (resource xh, string logfile [, int loglevel])

Benennen einer Log-Datei, in welcher die Fehlermeldungen des XSLT Prozessors abgelegt werden sollen.

xslt_output_begintransform (4.0.3 - 4.0.6 only)

unknown

unknown **xslt_output_begintransform** (unknown)

Zu dieser Funktion sind keine Prototyp Definitionen vorhanden.

xslt_output_endtransform (4.0.3 - 4.0.6 only)

unknown

unknown **xslt_output_endtransform** (unknown)

Zu dieser Funktion sind keine Prototyp Definitionen vorhanden.

xslt_output_process (unknown)

unknown

unknown **xslt_process** (unknown)

Zu dieser Funktion sind keine Prototyp Definitionen vorhanden.

xslt_run (4.0.3 - 4.0.6 only)

Ein XSLT Stylesheet einer XML Datei zuweisen.

bool **xslt_run** (resource xh, string xslt_file, string xml_data_file [, string result [, array xslt_params [, array xslt_args]]])

Auf eine XML Datei (xml_data_file) ein XSL Stylesheet aus einer Datei (xslt_file stylesheet) anwenden. Das Stylesheet hat Zugang zu xslt_params und der Prozessor wird mit xslt_args gestartet. Das Resultat der XSLT Transformation wird in einem benannten Speicher abgelegt (default is "/_result").

xslt_set_sax_handler (4.0.3 - 4.0.6 only)

Setzen des SAX handlers für einen XSLT Prozessor

bool **xslt_set_sax_handler** (resource xh, array handlers)

Setzen des SAX handlers für einen XSLT Prozessor, welcher durch den handle xh gegeben ist.

xslt_transform (4.0.3 - 4.0.6 only)

unknown

unknown **xslt_transform** (unknown)

Zu dieser Funktion sind keine Prototyp Definitionen vorhanden.

CV. YAZ functions

Introduction

This extension offers a PHP interface to the YAZ toolkit that implements the Z39.50 protocol for information retrieval. With this extension you can easily implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

YAZ is available at http://www.indexdata.dk/yaz/. You can find news information, example scripts, etc. for this extension at http://www.indexdata.dk/phpyaz/.

The module hides most of the complexity of Z39.50 so it should be fairly easy to use. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and initialize phase steps in most cases.

Installation

Compile YAZ and install it. Build PHP with your favourite modules and add option `--with-yaz`. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd ../php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

Example

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

Beispiel 1. Parallel searching using YAZ()

The script below demonstrates the parallel searching feature of the API. When invoked with no arguments it prints a query form; else (arguments are supplied) it searches the targets as given in in array host.

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i < $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

yaz_addinfo (PHP 4)

Returns additional error information

int **yaz_addinfo** (int id)

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_close (PHP 4)

Closes a YAZ connection

int **yaz_close** (int id)

Closes the Z-association given by *id*. The *id* is a target ID as returned by a previous yaz_connect() command.

yaz_connect (PHP 4)

Prepares for a connection and Z-association to a Z39.50 target.

int **yaz_connect** (string zurl [, mixed options])

This function returns a positive ID on success; zero on failure.

yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when yaz_wait() is called.

If the second argument, *options*, is given as a string it is treated as the Z39.50 V2 authentication string (OpenAuth).

If *options* is given as an array the contents of the array serves as options. Note that array options are only supported for PHP 4.1.0 and later.

yaz_connect() options

user

Username for authentication.

group

Group for authentication.

password

Password for authentication.

cookie

Cookie for session (YAZ proxy).

proxy

Proxy for connection (YAZ proxy).

persistent

A boolean. If `TRUE` the connection is persistent; If `FALSE` the connection is not persistent. By default connections are persistent.

piggyback

A boolean. If `TRUE` piggyback is enabled for searches; If `FALSE` piggyback is disabled. By default piggyback is enabled. Enabling piggyback is more efficient and usually saves a network-round-trip for first time fetches of records. However, a few Z39.50 targets doesn't support piggyback or they ignore element set names. For those, piggyback should be disabled.

yaz_errno (PHP 4)

Returns error number

int **yaz_errno** (int id)

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors (such as when the target closed connection, etc).

yaz_errno() should be called after network activity for each target - (after yaz_wait() returns) to determine the success or failure of the last operation (e.g. search).

yaz_error (PHP 4)

Returns error description

string **yaz_error** (int id)

Returns error message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns an english text message corresponding to the last error number as returned by yaz_errno().

yaz_hits (PHP 4)

Returns number of hits for last search

int **yaz_hits** (int id)

yaz_hits() returns number of hits for last search.

yaz_element (PHP 4)

Specifies Element-Set Name for retrieval

int **yaz_element** (int id, string elementset)

This function is used in conjunction with yaz_search() and yaz_present() to specify the element set name for records to be retrieved. Most servers support `F` (full) and `B` (brief).

Returns `TRUE` on success; `FALSE` on error.

yaz_database (PHP 4 >= 4.0.6)

Specifies the databases within a session

int **yaz_database** (int id, string databases)

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to yaz_connect(). Multiple databases are separated by a plus sign `+`.

This function allows you to use different sets of databases within a session.

Returns `TRUE` on success; `FALSE` on error.

yaz_range (PHP 4)

Specifies the maximum number of records to retrieve

int **yaz_range** (int id, int start, int number)

This function is used in conjunction with yaz_search() to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only yaz_search()) start is set to 1 and number is set to 10.

Returns `TRUE` on success; `FALSE` on error.

yaz_record (PHP 4)

Returns a record

int **yaz_record** (int id, int pos, string type)

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array" the record is returned as an array representation (for structured records).

yaz_search (PHP 4)

Prepares for a search

int **yaz_search** (int id, string type, string query)

yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument specifies a Type-1 query (RPN). Like yaz_connect() this function is non-blocking and only prepares for a search to be executed later when yaz_wait() is called.

The RPN query is a textual represenation of the Type-1 query as defined by the Z39.50 standard. However, in the text representation as used by YAZ a prefix notation is used, that is the operater precedes the operands. The query string is a sequence of tokens where white space is ignored is ignored unless surrounded by double quotes. Tokens beginning with an at-character (@) are considered operators, otherwise they are treated as search terms.

Tabelle 1. RPN Operators

Syntax	Description
`@and query1 query2`	intersection of query1 and query2
`@or query1 query2`	union of query1 and query2
`@not query1 query2`	query1 and not query2
`@set name`	result set reference
`@attrset set query`	specifies attribute-set for query. This construction is only allowed once - in the beginning of the whole query
`@attr set type=value query`	applies attribute to query. The type and value are integers specifying the attribute-type and attribute-value respectively. The set, if given, specifies the attribute-set.

Beispiel 1. Query Examples

Query

`computer`
matches documents where "computer" occur. No attributes are specified.

The Query

`"donald knuth"`
matches documents where "donald knuth" occur.

For the query

`@attr 1=4 art`
attribute type is 1 (Bib-1 use), attribute value is 4 Title), so this should match documents where `art` occur in the title.

Another more complex one:

```
@attrset gils @and @attr 1=4 art @attr 1=1003 "donald knuth"
```

The query as a whole uses the GILS attributeset. The query matches documents where `art` occur in the title and in which `donald knuth` occur in the author.

yaz_present (PHP 4 >= 4.0.5)

Prepares for retrieval (Z39.50 present).

int **yaz_present** (void)

This function prepares for retrieval of records after a successful search. The yaz_range() should be called prior to this function to specify the range of records to be retrieved.

yaz_syntax (PHP 4)

Specifies the preferred record syntax for retrieval.

int **yaz_syntax** (int id, string syntax)

The syntax is specified as an OID (Object Identifier) in a raw dot-notation (like `1.2.840.10003.5.10`) or as one of the known registered record syntaxes (sutrs, usmarc, grs1, xml, etc.). This function is used in conjunction with yaz_search() and yaz_present() to specify the preferred record syntax for retrieval.

yaz_scan (PHP 4 >= 4.0.5)

Prepares for a scan

int **yaz_scan** (int id, string type, string startterm [, array flags])

This function prepares for a Z39.50 Scan Request. Argument *`id`* specifies target ID. Starting term point for the scan is given by *`startterm`*. The form in which is the starting term is specified is given by *`type`*. Currently type `rpn` is supported. The optional *`flags`* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: `number` (number of terms requested), `position` (preferred position of term) and `stepSize` (preferred step size). To actually tranfer the Scan Request to the target and receive the Scan Response, yaz_wait() must be called. Upon completion of yaz_wait() call yaz_error() and yaz_scan_result() to handle the response.

The syntax of *`startterm`* is similar to the RPN query as described in yaz_search(). The startterm consists of zero or more `@attr`-operator specifications, then followed by exactly one token.

Beispiel 1. PHP function that scans titles

```
function scan_titles($id, $starterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $starterm);
```

```
    yaz_wait();
    $errno = yaz_errno($id);
    if ($errno == 0) {
      $ar = yaz_scan_result($id,&$options);
      echo 'Scan ok; ';
      $ar = yaz_scan_result($id, &$options);
      while(list($key,$val)=each($options)) {
         echo "$key = $val  ";
      }
      echo '<br><table><tr><td>';
      while(list($key,list($k, $term, $tcount))=each($ar)) {
        if (empty($k)) continue;
        echo "<tr><td>$term</td><td>";
        echo $tcount;
        echo "</td></tr>";
      }
      echo '</table>';
    } else {
      echo "Scan failed. Error: " . yaz_error($id) . "<br>";
    }
  }
```

yaz_scan_result (PHP 4 >= 4.0.5)

Returns Scan Response result

array **yaz_scan_result** (int id [, array & result])

Given a target ID this function returns and array with terms as received from the target in the last Scan Response. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where first item is term, second item is result-count. If the *result* is given it will be modified to hold additional information taken from the Scan Response: `number` (number of entries returned), `stepsize` (Step-size), `position` (position of term), `status` (Scan Status).

yaz_ccl_conf (PHP 4 >= 4.0.5)

Configure CCL parser

int **yaz_ccl_conf** (int id, array config)

This function configures the CCL query parser for a target with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the yaz_ccl_parse() function. Each index of the array *config* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (=). Each pair is separated by white space.

Beispiel 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: `ti`, `au` and `isbn`. Each field is mapped to their BIB-1 equivalent. It is assumed that variable `$id` is a target ID.

```
$field["ti"] = "1=4";
$field["au"] = "1=1";
$field["isbn"] = "1=7";
yaz_ccl_conf($id,$field);
```

yaz_ccl_parse (PHP 4 >= 4.0.5)

Invoke CCL Parser

int **yaz_ccl_parse** (int id, string query, array & result)

This function invokes the CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the yaz_search() function to perform a search. To define a set of valid CCL fields call yaz_ccl_conf() prior to this function. If the supplied *`query`* was successfully converted to RPN, this function returns `TRUE`, and the index `rpn` of the supplied array *`result`* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns `FALSE` and three indexes are set in the resulting array to indicate the cause of failure: `errorcode`CCL error code (integer), `errorstring`CCL error string, and `errorpos`approximate position in query of failure (integer is character position).

yaz_itemorder (PHP 4 >= 4.0.5)

Prepares for Z39.50 Item Order with an ILL-Request package

int **yaz_itemorder** (array args)

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this (http://www.nlc-bnc.ca/iso/ill/stanprf.htm) and the specification (http://www.nlc-bnc.ca/iso/ill/document/standard/z-ill-1a.pdf). The args parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier

transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number

item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_wait (PHP 4)

Wait for Z39.50 requests to complete

int **yaz_wait** ([array options])

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions yaz_connect(), yaz_search(), yaz_present(), yaz_scan() and yaz_itemorder(). **yaz_wait()** returns when all targets have either completed all requests or aborted (in case of errors).

If the `options` array is given that holds options that change the behaviour of **yaz_wait()**.

`timeout`

Sets timeout in seconds. If a target hasn't responded within the timeout it is considered dead and **yaz_wait()** returns. The default value for timeout is 15 seconds.

yaz_sort (PHP 4 >= 4.1.0)

Sets sorting criteria

int **yaz_sort** (int id, string criteria)

This function sets sorting criteria and enables Z39.50 Sort. Use this function together with yaz_search() or yaz_present(). Using this function alone doesn't have any effect. If used in conjunction with yaz_search() a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present. The *`criteria`* takes the form

`field1 flags1 field2 flags2` ...

where field1 specifies primary attributes for sort, field2 seconds, etc.. The field specifies either numerical attribute combinations consisting of type=value pairs separated by comma (e.g. `1=4,2=1`) ; or the field may specify a plain string criteria (e.g. `title`. The flags is a sequnce of the following characters which may not be separated by any white space.

Sort Flags

`a`

Sort ascending

`d`

Sort descending

`i`

Case insensitive sorting

`s`

Case sensitive sorting

Beispiel 1. Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you'd use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you'd use:

```
1=4 ia 1=1003 sa
```

CVI. YP/NIS Funktionen

NIS (früher unter dem Namen Yellow Pages bekannt) erlauben das netzwerkweite Management von wichtigen administrativen Dateien (z.B der Passwort-Datei. Für weitere Informationen zu NIS empfehlen wir die NIS Manpage und die Einführung zu YP/NIS (http://www.desy.de/~sieversm/ypdoku/ypdoku/ypdoku.html) Des weiteren ist ein empfehlenswertes Buch mit dem Titel Managing NFS and NIS (http://www.oreilly.com/catalog/nfs/noframes.html) von Hal Stern erhältlich.

Um die NIS-Funktionen mit PHP nutzen zu können, muß PHP vor der Compilierung mit den Optionen `--with-yp`(PHP 3) oder `--enable-yp`(PHP 4) konfiguriert werden.

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt die voreingestellte NIS-Domäne der aktuellen Maschine zurück.

int **yp_get_default_domain** (void)

yp_get_default_domain() gibt die voreingestellt Domäne des aktuellen Knotens oder FALSE zurück. Kann für erfolgreiche NIS Aufrufe als der Parameter domain genutzt werden.

Eine NIS-Domäne kann als Gruppe von NIS-Maps beschrieben werden. Jeder Host, der Informationen abrufen möchte, verbindet sich selbst zu einer bestimmten Domäne. Weitere Informationen hierzu finden sie in den oben genannten Quellen.

Beispiel 1. Beispiel für die voreingestellte Domäne

```
<?php
$domain = yp_get_default_domain();
echo "Default NIS domain is: " . $domain;
?>
```

yp_order (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt die Ordnungsnummer einer Map zurück

int **yp_order** (string domain, string map)

yp_order() gibt die Nummer einer Map in der Reihenfolge der NIS Maps oder FALSE zurück.

Beispiel 1. Beispiel für die NIS Reihenfolge

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Order number for this map is: " . $number;
?>
```

See also yp-get-default-domain().

yp_master (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt den Namen des Hauptservers für eine NIS-Map zurück

string **yp_master** (string domain, string map)

yp_master() gibt den Namen des Hauptservers (Master) für eine NIS-Map zurück.

Beispiel 1. Beispiel für den NIS Master

```
<?php
$number = yp_master ($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

Siehe auch yp-get-default-domain().

yp_match (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt die übereinstimmende Zeile zurück

string **yp_match** (string domain, string map, string key)

yp_match() gibt den Wert zurück, der mit dem übergebenen Schlüssel der angegebenen Map verbunden ist, bei einem Fehler `FALSE`. Der Schlüssel muss eindeutig sein.

Beispiel 1. Beispiel für den übereinstimmenden NIS-Eintrag

```
<?php
$entry = yp_match ($domain, "passwd.byname", "joe");
echo "Matched entry is: " . $entry;
?>
```

In diesem Fall wäre folgendes möglich: joe:##joe:11111:100:Joe User:/home/j/joe:/usr/local/bin/bash

See also yp-get-default-domain()

yp_first (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt das erste Schlüssel-Wert-Paar der angegebenen Map zurück

array **yp_first** (string domain, string map)

yp_first() gibt das erste Schlüssel-Wert-Paar der angegebenen Map zurück, andernfalls `FALSE`.

Beispiel 1. Beispiel für den ersten NIS-Eintrag

```
<?php
$entry = yp_first($domain, "passwd.byname");
$key = $entry ["key"];
$value = $entry ["value"];
echo "First entry in this map has key " . $key . " and value " . $value;
?>
```

See also yp-get-default-domain()

yp_next (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gibt das nächste Schlüssel-Wert-Paar in der angegebenen Map zurück.

array **yp_next** (string domain, string map, string key)

yp_next() gibt den das nächste Schlüssel-Wert-Paar in der angegebene Map zurück, nachdem der Schlüssel übergeben wurde, bei einem Fehler `FALSE`.

Beispiel 1. Beispiel für den nächsten NIS-Eintrag

```
<?php
$entry = yp_next ($domain, "passwd.byname", "joe");

if (!$entry) {
echo "No more entries found\n";

}

$key = key ($entry);

echo "The next entry after joe has key " . $key
      . " and value " . $entry[$key];
?>
```

Siehe auch yp-get-default-domain().

CVII. ZIP Funktionen (Lesezugriff)

Dieses Modul benutzt die Funktionen der ZZIPlib (http://zziplib.sourceforge.net/) Bibliothek von Guido Draheim um ZIP komprimierte Archive und die darin enthaltenen Dateien zu lesen.

Zu beachten ist, dass ZZIPlib nur eine Untermenge der Funktionen zur Verfügung stellt, die in einer vollständigen Implementation des ZIP Algorithmus vorhanden sind. Zur Erstellung eines ZIP Archivs muss man sich eines der üblichen ZIP Programme bedienen.

Die ZIP Unterstützung ist standardmäßig nicht aktiviert. Um die ZIP Funktionen nutzen zu können, muss PHP mit der option --with-zip kompiliert werden. Das ZIP Modul benötigt ZZIPlib version >= 0.10.6.

> **Anmerkung:** Zip Unterstützung für PHP vor Version 4.1.0 ist als experimentell anzusehen. Dieser Abschnitt beschreibt die ZIP Unterstützung, wie sie für PHP ab Version 4.1.0 existiert.

Beispiel zur Verwendung

Dieses Beispiel öffnet ein ein ZIP Archiv, liest jede Datei innerhalb des Archivs und gibt den Inhalt zurück. Das `test2.zip` Archiv, das in diesem Beispiel benutzt wird, wird mit der Original Distribution der ZZIPlib Bibliothek mitgeliefert.

Beispiel 1. Beispiel zur Verwendung der Zip Funktion

```
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {

    while ($zip_entry = zip_read($zip)) {
        echo "Name:               " . zip_entry_name($zip_entry) . "\n";
        echo "Actual Filesize:    " . zip_entry_filesize($zip_entry) . "\n";
        echo "Compressed Size:    " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Compression Method: " . zip_entry_compressionmethod($zip_entry) . "\n";

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "File Contents:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";

            zip_entry_close($zip_entry);
        }
        echo "\n";

    }

    zip_close($zip);

}

?>
```

zip_open (PHP 4 >= 4.1.0)

Öffnet ein ZIP Archiv

resource **zip_open** (string filename)

Diese Funktion öffnet ein ZIP Archiv für den Lesezugriff. Der Parameter `filename` ist der Name der zu öffnenden ZIP Datei.

Gibt einen Dateizeiger zurück, der in den Funktionen zip_read() und zip_close() benutzt wird. Existiert die Datei `filename` nicht wird `FALSE` zurückgegeben.

Siehe auch zip_read() und zip_close().

zip_read (PHP 4 >= 4.1.0)

Liest den nächsten Eintrag innerhalb des ZIP Archivs

resource **zip_read** (resource zip)

Liest den nächsten Eintrag innerhalb des ZIP Archivs. Der Parameter `zip` muss ein Dateizeiger sein, der vorher durch zip_open() zurückgegeben wurde.

Gibt einen Verzeichniseintrag zur späteren Benutzung mit den **zip_entry_...()** Funktionen zurück.

Siehe auch zip_open(), zip_close(), zip_entry_open() und zip_entry_read().

zip_close (PHP 4 >= 4.1.0)

Schließt ein ZIP Archiv

void **zip_close** (resource zip)

Schließt ein ZIP Archiv. Der Parameter `zip` muss ein Dateizeiger sein, der vorher durch zip_open() zurückgegeben wurde.

Diese Funktion hat keinen Rückgabewert.

Siehe auch zip_open() und zip_read().

zip_entry_close (PHP 4 >= 4.1.0)

Schließt einen Verzeichniseintrag

void **zip_entry_close** (resource zip_entry)

Schließt einen Verzeichniseintrag, der durch den Parameter `zip_entry` identifiziert wird. Dieser Parameter muss ein gültiger Verzeichniseintrag sein, der vorher durch zip_entry_open() geöffnet wurde.

Diese Funktion hat keinen Rückgabewert.

Siehe auch zip_entry_open() und zip_entry_read().

zip_entry_compressedsize (PHP 4 >= 4.1.0)

Gibt die komprimierte Größe eines Verzeichniseintrages zurück

int **zip_entry_compressedsize** (resource zip_entry)

Gibt die komprimierte Größe des Verzeichniseintrages zurück, der durch `zip_entry` spezifiziert wird. Der Parameter `zip_entry` muss ein gültiger Verzeichniseintrag sein, der durch zip_read() zurückgegeben wurde.

Siehe auch zip_open() und zip_read().

zip_entry_compressionmethod (PHP 4 >= 4.1.0)

Gibt die Komprimierungsmethode eines Verzeichniseintrages zurück.

string **zip_entry_compressionmethod** (resource zip_entry)

Gibt die Komprimierungsmethode eines Verzeichniseintrages zurück, der durch `zip_entry` spezifiziert wird. Der Parameter `zip_entry` muss ein gültiger Verzeichniseintrag sein, der durch zip_read() zurückgegeben wurde.

Siehe auch zip_open() und zip_read().

zip_entry_filesize (PHP 4 >= 4.1.0)

Gibt die Größe eines Verzeichniseintrages zurück

int **zip_entry_filesize** (resource zip_entry)

Gibt die Größe eines Verzeichniseintrages zurück, der durch `zip_entry` spezifiziert wird. Der Parameter `zip_entry` muss ein gültiger Verzeichniseintrag sein, der durch zip_read() zurückgegeben wurde.

Siehe auch zip_open() und zip_read().

zip_entry_name (PHP 4 >= 4.1.0)

Gibt den Namen eines Verzeichniseintrages zurück

string **zip_entry_name** (resource zip_entry)

Gibt den Namen eines Verzeichniseintrages zurück, der durch den Parameter `zip_entry` spezifiziert wurde. Der Parameter `zip_entry` muss ein gültiger Verzeichniseintrag sein, der durch zip_read() zurückgegeben wurde.

Siehe auch zip_open() und zip_read().

zip_entry_open (PHP 4 >= 4.1.0)

Öffnet einen Verzeichniseintrag für den Lesezugriff

bool **zip_entry_open** (resource zip, resource zip_entry [, string mode])

Öffnet einen Verzeichniseintrag für den Lesezugriff. Der Parameter *zip* muss ein Dateizeiger sein, der vorher durch zip_open() zurückgegeben wurde. Der Parameter *zip_entry* ist ein Verzeichniseintrag, der durch zip_read() zurückgegeben wurde. Der optionale Parameter *mode* kann einer der in der Dokumentation zu fopen() beschriebenen Werte sein.

Anmerkung: Gegenwärtig wird der Parameter *mode* ignoriert und implizit auf `"rb"` gesetzt. Dies erfolgt aufgrund des ausschließlichen Lesezugriffs der aktuellen ZIP Unterstüzung für PHP. In der Dokumentation zu fopen() finden sich weitere Erklärungen zu diesem Parameter.

Gibt bei Erfolg `TRUE` zurück, andernfalls `FALSE`.

Anmerkung: Im Gegensatz zu fopen() und vergleichbaren Funktionen zeigt der Rückgabewert von **zip_entry_open()** nur das Gelingen an und wird nicht für weitere Operationen benötigt.

Siehe auch zip_entry_read() und zip_entry_close().

zip_entry_read (PHP 4 >= 4.1.0)

Liest einen geöffneten Verzeichniseintrag aus

string **zip_entry_read** (resource zip_entry [, int length])

Liest bis zu *length* Bytes von einem geöffneten Verzeichniseintrag aus. Wenn der Parameter *length* nicht angegeben wurde, wird versucht die ersten 1024 Bytes auszulesen. Der Parameter *zip_entry* muss ein gültiger Verzeichniseintrag sein, der durch zip_read() zurückgegeben wurde.

Anmerkung: Der Parameter *length* bezieht sich auf die unkomprimierte Größe.

Gibt die ausgelesenen Daten zurück oder `FALSE`, wenn das Ende der Datei erreicht wurde.

Siehe auch zip_entry_open(), zip_entry_close() und zip_entry_filesize().

CVIII. Zlib Compression Functions

This module uses the functions of zlib (http://www.gzip.org/zlib/) by Jean-loup Gailly and Mark Adler to transparently read and write gzip (.gz) compressed files. You have to use a zlib version >= 1.0.9 with this module.

This module contains versions of most of the filesystem functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

> **Anmerkung:** The current CVS version 4.0.4-dev introduces a fopen-wrapper for .gz-files, so that you can use a special 'zlib:' URL to access compressed files transparently using the normal f*() file access functions if you prepend the filename or path with a 'zlib:' prefix when calling fopen().
>
> This feature requires a C runtime library that provides the `fopencookie()` function. To my current knowledge the GNU libc is the only library that provides this feature.

Small code example

Opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

Beispiel 1. Small Zlib Example

```
<?php

$filename = tempnam ('/tmp', 'zlibtest').'.gz';
print "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, "w9");

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

// open file for reading
$zp = gzopen($filename, "r");

// read 3 char
print gzread($zp, 3);

// output until end of the file and close it.
gzpassthru($zp);

print "\n";

// open file and print content (the 2nd time).
if (readgzfile($filename) != strlen($s)) {
        echo "Error with zlib functions!";
}
unlink($filename);
```

```
print "</pre>\n</h1></body>\n</html>\n";

?>
```

gzclose (PHP 3, PHP 4 >= 4.0.0)

Close an open gz-file pointer

int **gzclose** (int zp)

The gz-file pointed to by zp is closed.

Returns TRUE on success and FALSE on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

gzeof (PHP 3, PHP 4 >= 4.0.0)

Test for end-of-file on a gz-file pointer

int **gzeof** (int zp)

Returns TRUE if the gz-file pointer is at EOF or an error occurs; otherwise returns FALSE.

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

gzfile (PHP 3, PHP 4 >= 4.0.0)

Read entire gz-file into an array

array **gzfile** (string filename [, int use_include_path])

Identical to readgzfile(), except that **gzfile()** returns the file in an array.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also readgzfile(), and gzopen().

gzgetc (PHP 3, PHP 4 >= 4.0.0)

Get character from gz-file pointer

string **gzgetc** (int zp)

Returns a string containing a single (uncompressed) character read from the file pointed to by zp. Returns FALSE on EOF (as does gzeof()).

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), and gzgets().

gzgets (PHP 3, PHP 4 >= 4.0.0)

Get line from file pointer

string **gzgets** (int zp, int length)

Returns a (uncompressed) string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), gzgetc(), and fgets().

gzgetss (PHP 3, PHP 4 >= 4.0.0)

Get line from gz-file pointer and strip HTML tags

string **gzgetss** (int zp, int length [, string allowable_tags])

Identical to gzgets(), except that **gzgetss()** attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Anmerkung: *Allowable_tags* was added in PHP 3.0.13, PHP4B3.

See also gzgets(), gzopen(), and strip_tags().

gzopen (PHP 3, PHP 4 >= 4.0.0)

Open gz-file

int **gzopen** (string filename, string mode [, int use_include_path])

Opens a gzip (.gz) file for reading or writing. The mode parameter is as in fopen() ("rb" or "wb") but can also include a compression level ("wb9") or a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman only compression as in "wb1h". (See the description of deflateInit2 in zlib.h for more information about the strategy parameter.)

gzopen() can be used to read a file which is not in gzip format; in this case gzread() will directly read from the file without decompression.

gzopen() returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns FALSE.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Beispiel 1. gzopen() Example

```
$fp = gzopen ("/tmp/file.gz", "r");
```

See also gzclose().

gzpassthru (PHP 3, PHP 4 >= 4.0.0)

Output all remaining data on a gz-file pointer

int **gzpassthru** (int zp)

Reads to EOF on the given gz-file pointer and writes the (uncompressed) results to standard output.

If an error occurs, returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

The gz-file is closed when **gzpassthru()** is done reading it (leaving *zp* useless).

gzputs (PHP 3, PHP 4 >= 4.0.0)

Write to a gz-file pointer

int **gzputs** (int zp, string str [, int length])

gzputs() is an alias to gzwrite(), and is identical in every way.

gzread (PHP 3, PHP 4 >= 4.0.0)

Binary-safe gz-file read

string **gzread** (int zp, int length)

gzread() reads up to *length* bytes from the gz-file pointer referenced by *zp*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen ($filename, "r");
$contents = gzread ($zd, 10000);
gzclose ($zd);
```

See also gzwrite(), gzopen(), gzgets(), gzgetss(), gzfile(), and gzpassthru().

gzrewind (PHP 3, PHP 4 >= 4.0.0)

Rewind the position of a gz-file pointer

int **gzrewind** (int zp)

Sets the file position indicator for zp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzseek() and gztell().

gzseek (PHP 3, PHP 4 >= 4.0.0)

Seek on a gz-file pointer

int **gzseek** (int zp, int offset)

Sets the file position indicator for the file referenced by zp to offset bytes into the file stream. Equivalent to calling (in C) `gzseek( zp, offset, SEEK_SET )`.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek then compresses a sequence of zeroes up to the new starting position.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

See also gztell() and gzrewind().

gztell (PHP 3, PHP 4 >= 4.0.0)

Tell gz-file pointer read/write position

int **gztell** (int zp)

Returns the position of the file pointer referenced by *zp*; i.e., its offset into the file stream.

If an error occurs, returns `FALSE`.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), gzseek() and gzrewind().

gzwrite (PHP 3, PHP 4 >= 4.0.0)

Binary-safe gz-file write

int **gzwrite** (int zp, string string [, int length])

gzwrite() writes the contents of *string* to the gz-file stream pointed to by *zp*. If the *length* argument is given, writing will stop after *length* (uncompressed) bytes have been written or the end of *string* is reached, whichever comes first.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

See also gzread(), gzopen(), and gzputs().

readgzfile (PHP 3, PHP 4 >= 4.0.0)

Output a gz-file

int **readgzfile** (string filename [, int use_include_path])

Reads a file, decompresses it and writes it to standard output.

readgzfile() can be used to read a file which is not in gzip format; in this case **readgzfile()** will directly read from the file without decompression.

Returns the number of (uncompressed) bytes read from the file. If an error occurs, `FALSE` is returned and unless the function was called as `@readgzfile`, an error message is printed.

The file *filename* will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also gzpassthru(), gzfile(), and gzopen().

gzcompress (PHP 4)

Compress a string

string **gzcompress** (string data [, int level])

This function returns a compressed version of the input *data* using the ZLIB data format, or `FALSE` if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the ZLIB compression algorithm see the document "ZLIB Compressed Data Format Specification version 3.3 (http://www.ietf.org/rfc/rfc1950.txt)" (RFC 1950).

> **Anmerkung:** This is *not* the same as gzip compression, which includes some header data. See gzencode() for gzip compression.

See also gzdeflate(), gzinflate(), gzuncompress(), gzencode().

gzuncompress (PHP 4)

Uncompress a deflated string

string **gzuncompress** (string data [, int length])

This function takes *data* compressed by gzcompress() and returns the original uncompressed data or FALSE on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also gzdeflate(), gzinflate(), gzcompress(), gzencode().

gzdeflate (PHP 4 >= 4.0.4)

Deflate a string

string **gzdeflate** (string data [, int level])

This function returns a compressed version of the input *data* using the DEFLATE data format, or FALSE if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the DEFLATE compression algorithm see the document "DEFLATE Compressed Data Format Specification version 1.3 (http://www.ietf.org/rfc/rfc1951.txt)" (RFC 1951).

See also gzinflate(), gzcompress(), gzuncompress(), gzencode().

gzinflate (PHP 4 >= 4.0.4)

Inflate a deflated string

string **gzinflate** (string data [, int length])

This function takes *data* compressed by gzdeflate() and returns the original uncompressed data or FALSE on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also gzcompress(). gzuncompress(), gzdeflate(), gzencode().

gzencode (PHP 4 >= 4.0.4)

Create a gzip compressed string

string **gzencode** (string data [, int level])

This function returns a compressed version of the input *data* compatible with the output of the **gzip** program, or FALSE if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression, if not given the default compression level will be 1.

The resulting data contains the appropriate headers and data structure to make a standard .gz file, e.g.:

Beispiel 1. Creating a gzip file

```
<?php
    $data = implode("", file("bigfile.txt"));
    $gzdata = gzencode($data, 9);
    $fp = fopen("bigfile.txt.gz", "w");
    fwrite($fp, $gzdata);
    fclose($fp);
?>
```

For more information on the GZIP file format, see the document: GZIP file format specification version 4.3 (http://www.ietf.org/rfc/rfc1952.txt) (RFC 1952).

See also gzcompress(). gzuncompress(), gzdeflate(), gzinflate().

Teil V. Extending PHP 4.0

Kapitel 24. Overview

"Extending PHP" is easier said than done. PHP has evolved to a full-fledged tool consisting of a few megabytes of source code, and to hack a system like this quite a few things have to be learned and considered. When structuring this chapter, we finally decided on the "learn by doing" approach. This is not the most scientific and professional approach, but the method that's the most fun and gives the best end results. In the following sections, you'll learn quickly how to get the most basic extensions to work almost instantly. After that, you'll learn about Zend's advanced API functionality. The alternative would have been to try to impart the functionality, design, tips, tricks, etc. as a whole, all at once, thus giving a complete look at the big picture before doing anything practical. Although this is the "better" method, as no dirty hacks have to be made, it can be very frustrating as well as energy- and time-consuming, which is why we've decided on the direct approach.

Note that even though this chapter tries to impart as much knowledge as possible about the inner workings of PHP, it's impossible to really give a complete guide to extending PHP that works 100% of the time in all cases. PHP is such a huge and complex package that its inner workings can only be understood if you make yourself familiar with it by practicing, so we encourage you to work with the source.

What Is Zend? and What Is PHP?

The name *Zend* refers to the language engine, PHP's core. The term *PHP* refers to the complete system as it appears from the outside. This might sound a bit confusing at first, but it's not that complicated (see Figure 9.1). To implement a Web script interpreter, you need three parts:

1. The *interpreter* part analyzes the input code, translates it, and executes it.
2. The *functionality* part implements the functionality of the language (its functions, etc.).
3. The *interface* part talks to the Web server, etc.

Zend takes part 1 completely and a bit of part 2; PHP takes parts 2 and 3. Together they form the complete PHP package. Zend itself really forms only the language core, implementing PHP at its very basics with some predefined functions. PHP contains all the modules that actually create the language's outstanding capabilities.

Abbildung 24-1. Figure 9.1. The internal structure of PHP.

The following sections discuss where PHP can be extended and how it's done.

Kapitel 25. Extension Possibilities

As shown in Figure 9.1 above, PHP can be extended primarily at three points: external modules, built-in modules, and the Zend engine. The following sections discuss these options.

External Modules

External modules can be loaded at script runtime using the function dl(). This function loads a shared object from disk and makes its functionality available to the script to which it’s being bound. After the script is terminated, the external module is discarded from memory. This method has both advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
External modules don’t require recompiling of PHP.	The shared objects need to be loaded every time a script is being executed (every hit), which is very slow.
The size of PHP remains small by "outsourcing" certain functionality.	External additional files clutter up the disk.
	Every script that wants to use an external module’s functionality has to specifically include a call to dl(), or the `extension` tag in `php.ini` needs to be modified (which is not always a suitable solution).

To sum up, external modules are great for third-party products, small additions to PHP that are rarely used, or just for testing purposes. To develop additional functionality quickly, external modules provide the best results. For frequent usage, larger implementations, and complex code, the disadvantages outweigh the advantages.

Third parties might consider using the `extension` tag in `php.ini` to create additional external modules to PHP. These external modules are completely detached from the main package, which is a very handy feature in commercial environments. Commercial distributors can simply ship disks or archives containing only their additional modules, without the need to create fixed and solid PHP binaries that don’t allow other modules to be bound to them.

Built-in Modules

Built-in modules are compiled directly into PHP and carried around with every PHP process; their functionality is instantly available to every script that’s being run. Like external modules, built-in modules have advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
No need to load the module specifically; the functionality is instantly available.	Changes to built-in modules require recompiling of PHP.
No external files clutter up the disk; everything resides in the PHP binary.	The PHP binary grows and consumes more memory.

Built-in modules are best when you have a solid library of functions that remains relatively unchanged, requires better than poor-to-average performance, or is used frequently by many scripts

on your site. The need to recompile PHP is quickly compensated by the benefit in speed and ease of use. However, built-in modules are not ideal when rapid development of small additions is required.

The Zend Engine

Of course, extensions can also be implemented directly in the Zend engine. This strategy is good if you need a change in the language behavior or require special functions to be built directly into the language core. In general, however, modifications to the Zend engine should be avoided. Changes here result in incompatibilities with the rest of the world, and hardly anyone will ever adapt to specially patched Zend engines. Modifications can’t be detached from the main PHP sources and are overridden with the next update using the "official" source repositories. Therefore, this method is generally considered bad practice and, due to its rarity, is not covered in this book.

Kapitel 26. Source Layout

Anmerkung: Prior to working through the rest of this chapter, you should retrieve clean, unmodified source trees of your favorite Web server. We're working with Apache (available at http://www.apache.org/) and, of course, with PHP (available at http://www.php.net/ - does it need to be said?).

Make sure that you can compile a working PHP environment by yourself! We won't go into this issue here, however, as you should already have this most basic ability when studying this chapter.

Before we start discussing code issues, you should familiarize yourself with the source tree to be able to quickly navigate through PHP's files. This is a must-have ability to implement and debug extensions.

After extracting the PHP archive, you'll see a directory layout similar to that in Figure 9.2.

Abbildung 26-1. Figure 9.2. Main directory layout of the PHP source tree.

The following table describes the contents of the major directories.

Directory	Contents
`php-4`	Main PHP source files and main header files; here you'll find all of PHP's API definitions, macros, etc. (imp
`ext`	Repository for dynamic and built-in modules; by default, these are the "official" PHP modules that have bee
`pear`	Directory for the PHP class repository. At the time of this writing, this is still in the design phase, but it's bei
`sapi`	Contains the code for the different server abstraction layers.
`TSRM`	Location of the "Thread Safe Resource Manager" (TSRM) for Zend and PHP.
`Zend`	Location of Zend's file; here you'll find all of Zend's API definitions, macros, etc. (important).

Discussing all the files included in the PHP package is beyond the scope of this chapter. However, you should take a close look at the following files:

- `php.h`, located in the main PHP directory. This file contains most of PHP's macro and API definitions.
- `zend.h`, located in the main Zend directory. This file contains most of Zend's macros and definitions.
- `zend_API.h`, also located in the Zend directory, which defines Zend's API.

You should also follow some sub-inclusions from these files; for example, the ones relating to the Zend executor, the PHP initialization file support, and such. After reading these files, take the time to navigate around the package a little to see the interdependencies of all files and modules - how they

relate to each other and especially how they make use of each other. This also helps you to adapt to the coding style in which PHP is authored. To extend PHP, you should quickly adapt to this style.

Extension Conventions

Zend is built using certain conventions; to avoid breaking its standards, you should follow the rules described in the following sections.

Macros

For almost every important task, Zend ships predefined macros that are extremely handy. The tables and figures in the following sections describe most of the basic functions, structures, and macros. The macro definitions can be found mainly in `zend.h` and `zend_API.h`. We suggest that you take a close look at these files after having studied this chapter. (Although you can go ahead and read them now, not everything will make sense to you yet.)

Memory Management

Resource management is a crucial issue, especially in server software. One of the most valuable resources is memory, and memory management should be handled with extreme care. Memory management has been partially abstracted in Zend, and you should stick to this abstraction for obvious reasons: Due to the abstraction, Zend gets full control over all memory allocations. Zend is able to determine whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks. The functions to be used are described in the following table:

Function	Description
emalloc()	Serves as replacement for **malloc()**.
efree()	Serves as replacement for **free()**.
estrdup()	Serves as replacement for **strdup()**.
estrndup()	Serves as replacement for **strndup()**. Faster than **estrdup()** and binary-safe. This is the recommended funct
ecalloc()	Serves as replacement for **calloc()**.
erealloc()	Serves as replacement for **realloc()**.

emalloc(), **estrdup()**, **estrndup()**, **ecalloc()**, and **erealloc()** allocate internal memory; **efree()** frees these previously allocated blocks. Memory handled by the **e*()** functions is considered local to the current process and is discarded as soon as the script executed by this process is terminated.

> **Warnung**
>
> To allocate resident memory that survives termination of the current script, you can use **malloc()** and **free()**. This should only be done with extreme care, however, and only in conjunction with demands of the Zend API; otherwise, you risk memory leaks.

Zend also features a thread-safe resource manager to provide better native support for multithreaded

Web servers. This requires you to allocate local structures for all of your global variables to allow concurrent threads to be run. Because the thread-safe mode of Zend was not finished back when this was written, it is not yet extensively covered here.

Directory and File Functions

The following directory and file functions should be used in Zend modules. They behave exactly like their C counterparts, but provide virtual working directory support on the thread level.

Zend Function	Regular C Function
V_GETCWD()	getcwd()
V_FOPEN()	fopen()
V_OPEN()	**open()**
V_CHDIR()	chdir()
V_GETWD()	**getwd()**
V_CHDIR_FILE()	Takes a file path as an argument and changes the current working directory to that file's directory.
V_STAT()	stat()
V_LSTAT()	lstat()

String Handling

Strings are handled a bit differently by the Zend engine than other values such as integers, Booleans, etc., which don't require additional memory allocation for storing their values. If you want to return a string from a function, introduce a new string variable to the symbol table, or do something similar, you have to make sure that the memory the string will be occupying has previously been allocated, using the aforementioned **e*()** functions for allocation. (This might not make much sense to you yet; just keep it somewhere in your head for now - we'll get back to it shortly.)

Complex Types

Complex types such as arrays and objects require different treatment. Zend features a single API for these types - they're stored using hash tables.

> **Anmerkung:** To reduce complexity in the following source examples, we're only working with simple types such as integers at first. A discussion about creating more advanced types follows later in this chapter.

Kapitel 27. PHP’s Automatic Build System

PHP 4 features an automatic build system that's very flexible. All modules reside in a subdirectory of the `ext` directory. In addition to its own sources, each module consists of an M4 file (for example, see http://www.gnu.org/manual/m4/html_mono/m4.html) for configuration and a Makefile.in file, which is responsible for compilation (the results of autoconf and automake; for example, see http://sourceware.cygnus.com/autoconf/autoconf.html and http://sourceware.cygnus.com/automake/automake.html).

Both files are generated automatically, along with `.cvsignore`, by a little shell script named `ext_skel` that resides in the `ext` directory. As argument it takes the name of the module that you want to create. The shell script then creates a directory of the same name, along with the appropriate `config.m4` and `Makefile.in` files.

Step by step, the process looks like this:

```
root@dev:/usr/local/src/php4/ext > ./ext_skel my_module
Creating directory
Creating basic files: config.m4 Makefile.in .cvsignore [done].
To use your new extension, you will have to execute the following steps:
    $ cd ..
    $ ./buildconf
    $ ./configure # (your extension is automatically enabled)
    $ vi ext/my_module/my_module.c
    $ make
Repeat the last two steps as often as necessary.
```

This instruction creates the aforementioned files. To include the new module in the automatic configuration and build process, you have to run `buildconf`, which regenerates the `configure` script by searching through the `ext` directory and including all found `config.m4` files.

Finally, running `configure` parses all configuration options and generates a makefile based on those options and the options you specify in `Makefile.in`.

Listing 9.1 shows the previously generated `Makefile.in`:

Abbildung 27-1. Listing 9.1. The default `Makefile.in`.

```
# $Id: Extending_Zend_Build.xml,v 1.5 2002/02/11 10:10:33 hholzgra Exp $
LTLIBRARY_NAME        = libmy_module.la
LTLIBRARY_SOURCES     = my_module.c
LTLIBRARY_SHARED_NAME = my_module.la include
$(top_srcdir)/build/dynlib.mk
```

There's not much to tell about this one: It contains the names of the input and output files. You could also specify build instructions for other files if your module is built from multiple source files.

The default `config.m4` shown in Listing 9.2 is a bit more complex:

Abbildung 27-2. Listing 9.2. The default `config.m4`.

```
dnl $Id: Extending_Zend_Build.xml,v 1.5 2002/02/11 10:10:33 hholzgra Exp $
dnl config.m4 for extension my_module
dnl don't forget to call PHP_EXTENSION(my_module)
dnl If your extension references something external, use with:
PHP_ARG_WITH(my_module, for my_module support,
dnl Make sure that the comment is aligned:
    [  --with-my_module             Include my_module support])
```

```
dnl Otherwise use enable:
PHP_ARG_ENABLE(my_module, whether to enable my_module support,
dnl Make sure that the comment is aligned:
    [  --enable-my_module           Enable my_module support])
if test "$PHP_MY_MODULE" != "no"; then
dnl Action..
PHP_EXTENSION(my_module, $ext_shared)
fi
```

If you're unfamiliar with M4 files (now is certainly a good time to get familiar), this might be a bit confusing at first; but it's actually quite easy.

Note: Everything prefixed with `dnl` is treated as a comment and is not parsed.

The `config.m4` file is responsible for parsing the command-line options passed to `configure` at configuration time. This means that it has to check for required external files and do similar configuration and setup tasks.

The default file creates two configuration directives in the `configure` script: `--with-my_module` and `--enable-my_module`. Use the first option when referring external files (such as the `--with-apache` directive that refers to the Apache directory). Use the second option when the user simply has to decide whether to enable your extension. Regardless of which option you use, you should uncomment the other, unnecessary one; that is, if you're using `--enable-my_module`, you should remove support for `--with-my_module`, and vice versa.

By default, the `config.m4` file created by `ext_skel` accepts both directives and automatically enables your extension. Enabling the extension is done by using the `PHP_EXTENSION` macro. To change the default behavior to include your module into the PHP binary when desired by the user (by explicitly specifying `--enable-my_module` or `--with-my_module`), change the test for `$PHP_MY_MODULE` to `== "yes"`:

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
    Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

This would require you to use `--enable-my_module` each time when reconfiguring and recompiling PHP.

Note: Be sure to run `buildconf` every time you change `config.m4`!

We'll go into more details on the M4 macros available to your configuration scripts later in this chapter. For now, we'll simply use the default files. The sample sources on the CD-ROM all have working `config.m4` files. To include them into the PHP build process, simply copy the source directories to your PHP `ext` directory, run `buildconf`, and then include the sample modules you want by using the appropriate `--enable-*` directives with `configure`.

Kapitel 28. Creating Extensions

We'll start with the creation of a very simple extension at first, which basically does nothing more than implement a function that returns the integer it receives as parameter. Listing 9.3 shows the source.

Abbildung 28-1. Listing 9.3. A simple extension.

```
/* include standard header */
#include "php.h"

/* declaration of functions to be exported */
ZEND_FUNCTION(first_module);

/* compiled function list so Zend knows what's in this module */
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* compiled module information */
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES
};

/* implement standard "stub" routine to introduce ourselves to Zend */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* implement function that is meant to be made available to PHP */
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parmeter);
}
```

This code contains a complete PHP module. We'll explain the source code in detail shortly, but first we'd like to discuss the build process. (This will allow the impatient to experiment before we dive into API discussions.)

Anmerkung: The example source makes use of some features introduced with the Zend version used in PHP 4.1.0 and above, it won't compile with older PHP 4.0.x versions.

Compiling Modules

There are basically two ways to compile modules:

- Use the provided "make" mechanism in the `ext` directory, which also allows building of dynamic loadable modules.
- Compile the sources manually.

The first method should definitely be favored, since, as of PHP 4.0, this has been standardized into a sophisticated build process. The fact that it is so sophisticated is also its drawback, unfortunately - it's hard to understand at first. We'll provide a more detailed introduction to this later in the chapter, but first let's work with the default files.

The second method is good for those who (for some reason) don't have the full PHP source tree available, don't have access to all files, or just like to juggle with their keyboard. These cases should be extremely rare, but for the sake of completeness we'll also describe this method.

Compiling Using Make. To compile the sample sources using the standard mechanism, copy all their subdirectories to the `ext` directory of your PHP source tree. Then run `buildconf`, which will create an updated `configure` script containing appropriate options for the new extension. By default, all the sample sources are disabled, so you don't have to fear breaking your build process.

After you run `buildconf`, `configure --help` shows the following additional modules:

```
--enable-array_experiments   BOOK: Enables array experiments
--enable-call_userland       BOOK: Enables userland module
--enable-cross_conversion    BOOK: Enables cross-conversion module
--enable-first_module        BOOK: Enables first module
--enable-infoprint           BOOK: Enables infoprint module
--enable-reference_test      BOOK: Enables reference test module
--enable-resource_test       BOOK: Enables resource test module
--enable-variable_creation   BOOK: Enables variable-creation module
```

The module shown earlier in Listing 9.3 can be enabled with `--enable-first_module` or `--enable-first_module=yes`.

Compiling Manually. To compile your modules manually, you need the following commands:

Action	Command
Compiling	cc -fpic -DCOMPILE_DL=1 -I/usr/local/include -I. -I.. -I../Zend -c -o `<your_object_file>` `<your_c_f`
Linking	cc -shared -L/usr/local/lib -rdynamic -o `<your_module_file>` `<your_object_file(s)>`

The command to compile the module simply instructs the compiler to generate position-independent code (`-fpic` shouldn't be omitted) and additionally defines the constant `COMPILE_DL` to tell the module code that it's compiled as a dynamically loadable module (the test module above checks for this; we'll discuss it shortly). After these options, it specifies a number of standard include paths that should be used as the minimal set to compile the source files.

Note: All include paths in the example are relative to the directory `ext`. If you're compiling from another directory, change the pathnames accordingly. Required items are the PHP directory, the `Zend` directory, and (if necessary), the directory in which your module resides.

The link command is also a plain vanilla command instructing linkage as a dynamic module.

You can include optimization options in the compilation command, although these have been omitted in this example (but some are included in the makefile template described in an earlier section).

Note: Compiling and linking manually as a static module into the PHP binary involves very long instructions and thus is not discussed here. (It's not very efficient to type all those commands.)

Kapitel 29. Using Extensions

Depending on the build process you selected, you should either end up with a new PHP binary to be linked into your Web server (or run as CGI), or with an .so (shared object) file. If you compiled the example file `first_module.c` as a shared object, your result file should be `first_module.so`. To use it, you first have to copy it to a place from which it's accessible to PHP. For a simple test procedure, you can copy it to your `htdocs` directory and try it with the source in Listing 9.4. If you compiled it into the PHP binary, omit the call to dl(), as the module's functionality is instantly available to your scripts.

> **Warnung**
>
> For security reasons, you *should not* put your dynamic modules into publicly accessible directories. Even though it *can* be done and it simplifies testing, you should put them into a separate directory in production environments.

Abbildung 29-1. Listing 9.4. A test file for first_module.so.

```
<?php

// remove next comment if necessary
// dl("first_module.so");

$param = 2;
$return = first_module($param);

print("We sent '$param' and got '$return'");

?>
```

Calling this PHP file in your Web browser should give you the output shown in Figure 9.3.

Abbildung 29-2. Figure 9.3. Output of first_module.php.

If required, the dynamic loadable module is loaded by calling the dl() function. This function looks for the specified shared object, loads it, and makes its functions available to PHP. The module exports the function **first_module()**, which accepts a single parameter, converts it to an integer, and returns the result of the conversion.

If you've gotten this far, congratulations! You just built your first extension to PHP.

Kapitel 30. Troubleshooting

Actually, not much troubleshooting can be done when compiling static or dynamic modules. The only problem that could arise is that the compiler will complain about missing definitions or something similar. In this case, make sure that all header files are available and that you specified their path correctly in the compilation command. To be sure that everything is located correctly, extract a clean PHP source tree and use the automatic build in the `ext` directory with the fresh files; this will guarantee a safe compilation environment. If this fails, try manual compilation.

PHP might also complain about missing functions in your module. (This shouldn't happen with the sample sources if you didn't modify them.) If the names of external functions you're trying to access from your module are misspelled, they'll remain as "unlinked symbols" in the symbol table. During dynamic loading and linkage by PHP, they won't resolve because of the typing errors - there are no corresponding symbols in the main binary. Look for incorrect declarations in your module file or incorrectly written external references. Note that this problem is specific to dynamic loadable modules; it doesn't occur with static modules. Errors in static modules show up at compile time.

Kapitel 31. Source Discussion

Now that you've got a safe build environment and you're able to include the modules into PHP files, it's time to discuss how everything works.

Module Structure

All PHP modules follow a common structure:

- Header file inclusions (to include all required macros, API definitions, etc.)
- C declaration of exported functions (required to declare the Zend function block)
- Declaration of the Zend function block
- Declaration of the Zend module block
- Implementation of **get_module()**
- Implementation of all exported functions

Header File Inclusions

The only header file you really have to include for your modules is `php.h`, located in the PHP directory. This file makes all macros and API definitions required to build new modules available to your code.

Tip: It's good practice to create a separate header file for your module that contains module-specific definitions. This header file should contain all the forward definitions for exported functions and include `php.h`. If you created your module using `ext_skel` you already have such a header file prepared.

Declaring Exported Functions

To declare functions that are to be exported (i.e., made available to PHP as new native functions), Zend provides a set of macros. A sample declaration looks like this:

```
ZEND_FUNCTION ( my_function );
```

`ZEND_FUNCTION` declares a new C function that complies with Zend's internal API. This means that the function is of type `void` and accepts `INTERNAL_FUNCTION_PARAMETERS` (another macro) as parameters. Additionally, it prefixes the function name with `zif`. The immediately expanded version of the above definitions would look like this:

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

Expanding `INTERNAL_FUNCTION_PARAMETERS` results in the following:

```
void zif_my_function( int ht
```

```
, zval * return_value
, zval * this_ptr
, int return_value_used
, zend_executor_globals * executor_globals
);
```

Since the interpreter and executor core have been separated from the main PHP package, a second API defining macros and function sets has evolved: the Zend API. As the Zend API now handles quite a few of the responsibilities that previously belonged to PHP, a lot of PHP functions have been reduced to macros aliasing to calls into the Zend API. The recommended practice is to use the Zend API wherever possible, as the old API is only preserved for compatibility reasons. For example, the types zval and pval are identical. zval is Zend's definition; pval is PHP's definition (actually, pval is an alias for zval now). As the macro `INTERNAL_FUNCTION_PARAMETERS` is a Zend macro, the above declaration contains zval. When writing code, you should always use zval to conform to the new Zend API.

The parameter list of this declaration is very important; you should keep these parameters in mind (see Table 9.1 for descriptions).

Abbildung 31-1. Table 9.1. Zend's Parameters to Functions Called from PHP

Parameter	Description
ht	The number of arguments passed to the Zend function. You should not touch this directly, but instea
return_value	This variable is used to pass any return values of your function back to PHP. Access to this variable
this_ptr	Using this variable, you can gain access to the object in which your function is contained, if it's use
return_value_used	This flag indicates whether an eventual return value from this function will actually be used by the c
executor_globals	This variable points to global settings of the Zend engine. You'll find this useful when creating new

Declaration of the Zend Function Block

Now that you have declared the functions to be exported, you also have to introduce them to Zend. Introducing the list of functions is done by using an array of zend_function_entry. This array consecutively contains all functions that are to be made available externally, with the function's name as it should appear in PHP and its name as defined in the C source. Internally, zend_function_entry is defined as shown in Listing 9.5.

Abbildung 31-2. Listing 9.5. Internal declaration of zend_function_entry.

```
typedef struct _zend_function_entry {
    char *fname;
    void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
    unsigned char *func_arg_types;
} zend_function_entry;
```

Entry	Description
fname	Denotes the function name as seen in PHP (for example, `fopen`, `mysql_connect`, or, in our example,
handler	Pointer to the C function responsible for handling calls to this function. For example, see the standard
func_arg_types	Allows you to mark certain parameters so that they're forced to be passed by reference. You usually sh

In the example above, the declaration looks like this:

```
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};
```

You can see that the last entry in the list always has to be `{NULL, NULL, NULL}`. This marker has to be set for Zend to know when the end of the list of exported functions is reached.

> **Anmerkung:** You *cannot* use the predefined macros for the end marker, as these would try to refer to a function named "NULL"!

The macro `ZEND_FE` (short for 'Zend Function Entry') simply expands to a structure entry in zend_function_entry. Note that these macros introduce a special naming scheme to your functions - your C functions will be prefixed with `zif_`, meaning that `ZEND_FE(first_module)` will refer to a C function **zif_first_module()**. If you want to mix macro usage with hand-coded entries (not a good practice), keep this in mind.

Tip: Compilation errors that refer to functions named **zif_*()** relate to functions defined with `ZEND_FE`.

Table 9.2 shows a list of all the macros that you can use to define functions.

Abbildung 31-3. Table 9.2. Macros for Defining Functions

Macro Name	Description
`ZEND_FE(name, arg_types)`	Defines a function entry of the name name in zend_functio
`ZEND_NAMED_FE(php_name, name, arg_types)`	Defines a function that will be available to PHP by the nam
`ZEND_FALIAS(name, alias, arg_types)`	Defines an alias named alias for name. arg_types needs to
`PHP_FE(name, arg_types)`	Old PHP API equivalent of `ZEND_FE`.
`PHP_NAMED_FE(runtime_name, name, arg_types)`	Old PHP API equivalent of `ZEND_NAMED_FE`.

Note: You can't use `ZEND_FE` in conjunction with `PHP_FUNCTION`, or `PHP_FE` in conjunction with `ZEND_FUNCTION`. However, it's perfectly legal to mix `ZEND_FE` and `ZEND_FUNCTION` with `PHP_FE` and `PHP_FUNCTION` when staying with the same macro set for each function to be declared. But mixing is *not* recommended; instead, you're advised to use the `ZEND_*` macros only.

Declaration of the Zend Module Block

This block is stored in the structure zend_module_entry and contains all necessary information to describe the contents of this module to Zend. You can see the internal definition of this module in Listing 9.6.

Abbildung 31-4. Internal declaration of zend_module_entry.

```
typedef struct _zend_module_entry zend_module_entry;

    struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    char *name;
    zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;
    int (*global_startup_func)(void);
    int (*global_shutdown_func)(void);

[ Rest of the structure is not interesting here ]

};
```

Entry	Description
size, zend_api, zend_debug and zts	Usually filled with the `"STANDARD_MODULE_HEADER"`, which fills these four me
name	Contains the module name (for example, `"File functions"`, `"Socket funct`
functions	Points to the Zend function block, discussed in the preceding section.
module_startup_func	This function is called once upon module initialization and can be used to do one-
module_shutdown_func	This function is called once upon module shutdown and can be used to do one-tim
request_startup_func	This function is called once upon every page request and can be used to do one-ti
request_shutdown_func	This function is called once after every page request and works as counterpart to
info_func	When phpinfo() is called in a script, Zend cycles through all loaded modules and
version	The version of the module. You can use `NO_VERSION_YET` if you don't want to g
Remaining structure elements	These are used internally and can be prefilled by using the macro `STANDARD_MOD`

In our example, this structure is implemented as follows:

```
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL, NULL, NULL, NULL, NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES,
```

```
};
```

This is basically the easiest and most minimal set of values you could ever use. The module name is set to `First Module`, then the function list is referenced, after which all startup and shutdown functions are marked as being unused.

For reference purposes, you can find a list of the macros involved in declared startup and shutdown functions in Table 9.3. These are not used in our basic example yet, but will be demonstrated later on. You should make use of these macros to declare your startup and shutdown functions, as these require special arguments to be passed (`INIT_FUNC_ARGS` and `SHUTDOWN_FUNC_ARGS`), which are automatically included into the function declaration when using the predefined macros. If you declare your functions manually and the PHP developers decide that a change in the argument list is necessary, you'll have to change your module sources to remain compatible.

Abbildung 31-5. Macros to Declare Startup and Shutdown Functions

Macro	Description
`ZEND_MINIT(module)`	Declares a function for module startup. The generated name will be `zend_minit_<mod`
`ZEND_MSHUTDOWN(module)`	Declares a function for module shutdown. The generated name will be `zend_mshutdow`
`ZEND_RINIT(module)`	Declares a function for request startup. The generated name will be `zend_rinit_<mod`
`ZEND_RSHUTDOWN(module)`	Declares a function for request shutdown. The generated name will be `zend_rshutdow`
`ZEND_MINFO(module)`	Declares a function for printing module information, used when phpinfo() is called. The

Creation of get_module()

This function is special to all dynamic loadable modules. Take a look at the creation via the `ZEND_GET_MODULE` macro first:

```
#if COMPILE_DL_FIRSTMOD
     ZEND_GET_MODULE(firstmod)
#endif
```

The function implementation is surrounded by a conditional compilation statement. This is needed since the function **get_module()** is only required if your module is built as a dynamic extension. By specifying a definition of `COMPILE_DL_FIRSTMOD` in the compiler command (see above for a discussion of the compilation instructions required to build a dynamic extension), you can instruct your module whether you intend to build it as a dynamic extension or as a built-in module. If you want a built-in module, the implementation of **get_module()** is simply left out.

get_module() is called by Zend at load time of the module. You can think of it as being invoked by the dl() call in your script. Its purpose is to pass the module information block back to Zend in order to inform the engine about the module contents.

If you don't implement a **get_module()** function in your dynamic loadable module, Zend will compliment you with an error message when trying to access it.

Implementation of All Exported Functions

Implementing the exported functions is the final step. The example function in `first_module` looks like this:

```
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAI-
LURE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

The function declaration is done using `ZEND_FUNCTION`, which corresponds to `ZEND_FE` in the function entry table (discussed earlier).

After the declaration, code for checking and retrieving the function's arguments, argument conversion, and return value generation follows (more on this later).

Summary

That's it, basically - there's nothing more to implementing PHP modules. Built-in modules are structured similarly to dynamic modules, so, equipped with the information presented in the previous sections, you'll be able to fight the odds when encountering PHP module source files.

Now, in the following sections, read on about how to make use of PHP's internals to build powerful extensions.

Kapitel 32. Accepting Arguments

One of the most important issues for language extensions is accepting and dealing with data passed via arguments. Most extensions are built to deal with specific input data (or require parameters to perform their specific actions), and function arguments are the only real way to exchange data between the PHP level and the C level. Of course, there's also the possibility of exchanging data using predefined global values (which is also discussed later), but this should be avoided by all means, as it's extremely bad practice.

PHP doesn't make use of any formal function declarations; this is why call syntax is always completely dynamic and never checked for errors. Checking for correct call syntax is left to the user code. For example, it's possible to call a function using only one argument at one time and four arguments the next time - both invocations are syntactically absolutely correct.

Determining the Number of Arguments

Since PHP doesn't have formal function definitions with support for call syntax checking, and since PHP features variable arguments, sometimes you need to find out with how many arguments your function has been called. You can use the `ZEND_NUM_ARGS` macro in this case. In previous versions of PHP, this macro retrieved the number of arguments with which the function has been called based on the function's hash table entry, ht, which is passed in the `INTERNAL_FUNCTION_PARAMETERS` list. As ht itself now contains the number of arguments that have been passed to the function, `ZEND_NUM_ARGS` has been stripped down to a dummy macro (see its definition in `zend_API.h`). But it's still good practice to use it, to remain compatible with future changes in the call interface. *Note:* The old PHP equivalent of this macro is `ARG_COUNT`.

The following code checks for the correct number of arguments:

```
if(ZEND_NUM_ARGS() != 2) WRONG_PARAM_COUNT;
```

If the function is not called with two arguments, it exits with an error message. The code snippet above makes use of the tool macro `WRONG_PARAM_COUNT`, which can be used to generate a standard error message (see Figure 9.4).

Abbildung 32-1. Figure 9.4. `WRONG_PARAM_COUNT` in action.

This macro prints a default error message and then returns to the caller. Its definition can also be found in `zend_API.h` and looks like this:

```
ZEND_API void wrong_param_count(void);

#define WRONG_PARAM_COUNT { wrong_param_count(); return; }
```

As you can see, it calls an internal function named **wrong_param_count()** that's responsible for printing the warning. For details on generating customized error messages, see the later section "Printing Information."

Retrieving Arguments

> **New parameter parsing API:** This chapter documents the new Zend parameter parsing API introduced by Andrei Zmievski. It was introduced in the development stage between PHP 4.0.6 and 4.1.0 .

Parsing parameters is a very common operation and it may get a bit tedious. It would also be nice to have standardized error checking and error messages. Since PHP 4.1.0, there is a way to do just that by using the new parameter parsing API. It greatly simplifies the process of receiving parameteres, but it has a drawback in that it can't be used for functions that expect variable number of parameters. But since the vast majority of functions do not fall into those categories, this parsing API is recommended as the new standard way.

The prototype for parameter parsing function looks like this:

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

The first argument to this function is supposed to be the number of actual parameters passed to your function, so `ZEND_NUM_ARGS()` can be used for that. The second parameter should always be `TSRMLS_CC` macro. The third argument is a string that specifies the number and types of arguments your function is expecting, similar to how printf format string specifies the number and format of the output values it should operate on. And finally the rest of the arguments are pointers to variables which should receive the values from the parameters.

zend_parse_parameters() also performs type conversions whenever possible, so that you always receive the data in the format you asked for. Any type of scalar can be converted to another one, but conversions between complex types (arrays, objects, and resources) and scalar types are not allowed.

If the parameters could be obtained successfully and there were no errors during type conversion, the function will return `SUCCESS`, otherwise it will return `FAILURE`. The function will output informative error messages, if the number of received parameters does not match the requested number, or if type conversion could not be performed.

Here are some sample error messages:

```
Warning - ini_get_all() requires at most 1 parameter, 2 given
Warning - wddx_deserialize() expects parameter 1 to be string, array given
```

Of course each error message is accompanied by the filename and line number on which it occurs.

Here is the full list of type specifiers:

- `l` - long
- `d` - double
- `s` - string (with possible null bytes) and its length
- `b` - boolean

- `r` - resource, stored in `zval*`
- `a` - array, stored in `zval*`
- `o` - object (of any class), stored in `zval*`
- `O` - object (of class specified by class entry), stored in `zval*`
- `z` - the actual `zval*`

The following characters also have a meaning in the specifier string:

- `|` - indicates that the remaining parameters are optional. The storage variables corresponding to these parameters should be initialized to default values by the extension, since they will not be touched by the parsing function if the parameters are not passed.
- `/` - the parsing function will call **SEPARATE_ZVAL_IF_NOT_REF()** on the parameter it follows, to provide a copy of the parameter, unless it's a reference.
- `!` - the parameter it follows can be of specified type or `NULL` (only applies to a, o, O, r, and z). If `NULL` value is passed by the user, the storage pointer will be set to `NULL`.

The best way to illustrate the usage of this function is through examples:

```
/* Gets a long, a string and its length, and a zval. */
long l;
char *s;
int s_len;
zval *param;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "lsz", &l, &s, &s_len, &param) == FAILURE) {
    return;
}

/* Gets an object of class specified by my_ce, and an optional double. */
zval *obj;
double d = 0.5;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "O|d", &obj, my_ce, &d) == FAILURE) {
    return;
}

/* Gets an object or null, and an array.
   If null is passed for object, obj will be set to NULL. */
zval *obj;
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "O!a", &obj, &arr) == FAI-
LURE) {
    return;
}

/* Gets a separated array. */
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a/", &arr) == FAILURE) {
    return;
}

/* Get only the first three parameters (useful for varargs functions). */
```

```
zval *z;
zend_bool b;
zval *r;
if (zend_parse_parameters(3, "zbr!", &z, &b, &r) == FAILURE) {
    return;
}
```

Note that in the last example we pass 3 for the number of received parameters, instead of **ZEND_NUM_ARGS()**. What this lets us do is receive the least number of parameters if our function expects a variable number of them. Of course, if you want to operate on the rest of the parameters, you will have to use **zend_get_parameters_array_ex()** to obtain them.

The parsing function has an extended version that allows for an additional flags argument that controls its actions.

```
int zend_parse_parameters_ex(int flags, int num_args TSRMLS_DC, char *type_spec, ...);
```

The only flag you can pass currently is `ZEND_PARSE_PARAMS_QUIET`, which instructs the function to not output any error messages during its operation. This is useful for functions that expect several sets of completely different arguments, but you will have to output your own error messages.

For example, here is how you would get either a set of three longs or a string:

```
long l1, l2, l3;
char *s;
if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                             ZEND_NUM_ARGS() TSRMLS_CC,
                             "lll", &l1, &l2, &l3) == SUCCESS) {
    /* manipulate longs */
} else if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                    ZEND_NUM_ARGS(), "s", &s, &s_len) == SUC-
CESS) {
    /* manipulate string */
} else {
    php_error(E_WARNING, "%s() takes either three long values or a string as argument",
              get_active_function_name(TSRMLS_C));
    return;
}
```

With all the abovementioned ways of receiving function parameters you should have a good handle on this process. For even more example, look through the source code for extensions that are shipped with PHP - they illustrate every conceivable situation.

Old way of retrieving arguments (deprecated)

> **Deprecated parameter parsing API:** This API is deprecated and superseded by the new ZEND parameter parsing API.

After having checked the number of arguments, you need to get access to the arguments themselves. This is done with the help of **zend_get_parameters_ex()**:

```
zval **parameter;

if(zend_get_parameters_ex(1, &parameter) != SUCCESS)
  WRONG_PARAM_COUNT;
```

All arguments are stored in a zval container, which needs to be pointed to *twice*. The snippet above tries to retrieve one argument and make it available to us via the parameter pointer.

zend_get_parameters_ex() accepts at least two arguments. The first argument is the number of arguments to retrieve (which should match the number of arguments with which the function has been called; this is why it's important to check for correct call syntax). The second argument (and all following arguments) are pointers to pointers to pointers to zvals. (Confusing, isn't it?) All these pointers are required because Zend works internally with **zval; to adjust a local **zval in our function,**zend_get_parameters_ex()** requires a pointer to it.

The return value of **zend_get_parameters_ex()** can either be `SUCCESS` or `FAILURE`, indicating (unsurprisingly) success or failure of the argument processing. A failure is most likely related to an incorrect number of arguments being specified, in which case you should exit with `WRONG_PARAM_COUNT`.

To retrieve more than one argument, you can use a similar snippet:

```
zval **param1, **param2, **param3, **param4;

if(zend_get_parameters_ex(4, &param1, &param2, &param3, &param4) != SUCCESS)
    WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() only checks whether you're trying to retrieve too many parameters. If the function is called with five arguments, but you're only retrieving three of them with **zend_get_parameters_ex()**, you won't get an error but will get the first three parameters instead. Subsequent calls of **zend_get_parameters_ex()** won't retrieve the remaining arguments, but will get the same arguments again.

Dealing with a Variable Number of Arguments/Optional Parameters

If your function is meant to accept a variable number of arguments, the snippets just described are sometimes suboptimal solutions. You have to create a line calling **zend_get_parameters_ex()** for every possible number of arguments, which is often unsatisfying.

For this case, you can use the function **zend_get_parameters_array_ex()**, which accepts the number of parameters to retrieve and an array in which to store them:

```
zval **parameter_array[4];

/* get the number of arguments */
argument_count = ZEND_NUM_ARGS();

/* see if it satisfies our minimal request (2 arguments) */
/* and our maximal acceptance (4 arguments) */
if(argument_count < 2 || argument_count > 5)
    WRONG_PARAM_COUNT;

/* argument count is correct, now retrieve arguments */
if(zend_get_parameters_array_ex(argument_count, parameter_array) != SUCCESS)
    WRONG_PARAM_COUNT;
```

First, the number of arguments is checked to make sure that it's in the accepted range. After that, **zend_get_parameters_array_ex()** is used to fill parameter_array with valid pointers to the argument values.

A very clever implementation of this can be found in the code handling PHP's fsockopen() located in `ext/standard/fsock.c`, as shown in Listing 9.7. Don't worry if you don't know all the functions used in this source yet; we'll get to them shortly.

Abbildung 32-2. Listing 9.7. PHP's implementation of variable arguments in fsockopen().

```
pval **args[5];
int *sock=emalloc(sizeof(int));
int *sockp;
int arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count > 5 || arg_count < 2 || zend_get_parameters_array_ex(arg_count,args)==FAILU
    CLOSE_SOCK(1);
    WRONG_PARAM_COUNT;
}

switch(arg_count) {
```

```
    case 5:
        convert_to_double_ex(args[4]);
        conv = (unsigned long) (Z_DVAL_P(args[4]) * 1000000.0);
        timeout.tv_sec = conv / 1000000;
        timeout.tv_usec = conv % 1000000;
        /* fall-through */
    case 4:
        if (!PZVAL_IS_REF(*args[3])) {
            php_error(E_WARNING,"error string argument to fsockopen not passed by referen
        }
        pval_copy_constructor(*args[3]);
        ZVAL_EMPTY_STRING(*args[3]);
        /* fall-through */
    case 3:
        if (!PZVAL_IS_REF(*args[2])) {
            php_error(E_WARNING,"error argument to fsockopen not passed by reference");
            return;
        }
        ZVAL_LONG(*args[2], 0);
        break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno = (unsigned short) Z_LVAL_P(args[1]);

key = emalloc(Z_STRLEN_P(args[0]) + 10);
```

fsockopen() accepts two, three, four, or five parameters. After the obligatory variable declarations, the function checks for the correct range of arguments. Then it uses a fall-through mechanism in a `switch()` statement to deal with all arguments. The `switch()` statement starts with the maximum number of arguments being passed (five). After that, it automatically processes the case of four arguments being passed, then three, by omitting the otherwise obligatory `break` keyword in all stages. After having processed the last case, it exits the `switch()` statement and does the minimal argument processing needed if the function is invoked with only two arguments.

This multiple-stage type of processing, similar to a stairway, allows convenient processing of a variable number of arguments.

Accessing Arguments

To access arguments, it’s necessary for each argument to have a clearly defined type. Again, PHP’s extremely dynamic nature introduces some quirks. Because PHP never does any kind of type checking, it’s possible for a caller to pass any kind of data to your functions, whether you want it or not. If you expect an integer, for example, the caller might pass an array, and vice versa - PHP simply won’t notice.

To work around this, you have to use a set of API functions to force a type conversion on every argument that’s being passed (see Table 9.4).

Note: All conversion functions expect a **zval as parameter.

Abbildung 32-3. Table 9.4. Argument Conversion Functions

Function	Description
convert_to_boolean_ex()	Forces conversion to a Boolean type. Boolean values remain untouched. Longs, d
convert_to_long_ex()	Forces conversion to a long, the default integer type. NULL values, Booleans, res
convert_to_double_ex()	Forces conversion to a double, the default floating-point type. NULL values, Bool
convert_to_string_ex()	Forces conversion to a string. Strings remain untouched. NULL values are conver
`convert_to_array_ex(value)`	Forces conversion to an array. Arrays remain untouched. Objects are converted to
`convert_to_object_ex(value)`	Forces conversion to an object. Objects remain untouched. NULL values are conv
`convert_to_null_ex(value)`	Forces the type to become a NULL value, meaning empty.

Anmerkung: You can find a demonstration of the behavior in `cross_conversion.php` on the accompanying CD-ROM. Figure 9.5 shows the output.

Abbildung 32-4. Figure 9.5. Cross-conversion behavior of PHP.

Using these functions on your arguments will ensure type safety for all data that's passed to you. If the supplied type doesn't match the required type, PHP forces dummy contents on the resulting value (empty strings, arrays, or objects, `0` for numeric values, `FALSE` for Booleans) to ensure a defined state.

Following is a quote from the sample module discussed previously, which makes use of the conversion functions:

```
zval **parameter;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &parameter) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

After retrieving the parameter pointer, the parameter value is converted to a long (an integer), which also forms the return value of this function. Understanding access to the contents of the value requires a short discussion of the zval type, whose definition is shown in Listing 9.8.

Abbildung 32-5. Listing 9.8. PHP/Zend zval type definition.

```
typedef pval zval;

typedef struct _zval_struct zval;

typedef union _zvalue_value {
long lval; /* long value */
double dval; /* double value */
struct {
char *val;
int len;
} str;
HashTable *ht; /* hash table value */
struct {
zend_class_entry *ce;
HashTable *properties;
} obj;
} zvalue_value;

struct _zval_struct {
/* Variable information */
zvalue_value value; /* value */
unsigned char type; /* active type */
unsigned char is_ref;
short refcount;
};
```

Actually, pval (defined in `php.h`) is only an alias of zval (defined in `zend.h`), which in turn refers to _zval_struct. This is a most interesting structure. _zval_struct is the "master" structure, containing the value structure, type, and reference information. The substructure zvalue_value is a union that contains the variable's contents. Depending on the variable's type, you'll have to access different members of this union. For a description of both structures, see Tables 9.5, 9.6, and 9.7.

Abbildung 32-6. Table 9.5. Zend zval Structure

Entry	Description
value	Union containing this variable's contents. See Table 9.6 for a description.
type	Contains this variable's type. For a list of available types, see Table 9.7.
is_ref	0 means that this variable is not a reference; 1 means that this variable is a reference to another variable.
refcount	The number of references that exist for this variable. For every new reference to the value stored in this variab

Abbildung 32-7. Table 9.6. Zend zvalue_value Structure

Entry	Description
lval	Use this property if the variable is of the type `IS_LONG`, `IS_BOOLEAN`, or `IS_RESOURCE`.
dval	Use this property if the variable is of the type `IS_DOUBLE`.
str	This structure can be used to access variables of the type `IS_STRING`. The member len contains the string length
ht	This entry points to the variable's hash table entry if the variable is an array.
obj	Use this property if the variable is of the type `IS_OBJECT`.

Abbildung 32-8. Table 9.7. Zend Variable Type Constants

Constant	Description
`IS_NULL`	Denotes a NULL (empty) value.
`IS_LONG`	A long (integer) value.
`IS_DOUBLE`	A double (floating point) value.
`IS_STRING`	A string.
`IS_ARRAY`	Denotes an array.
`IS_OBJECT`	An object.
`IS_BOOL`	A Boolean value.
`IS_RESOURCE`	A resource (for a discussion of resources, see the appropriate section below).
`IS_CONSTANT`	A constant (defined) value.

To access a long you access zval.value.lval, to access a double you use zval.value.dval, and so on. Because all values are stored in a union, trying to access data with incorrect union members results in meaningless output.

Accessing arrays and objects is a bit more complicated and is discussed later.

Dealing with Arguments Passed by Reference

If your function accepts arguments passed by reference that you intend to modify, you need to take some precautions.

What we didn't say yet is that under the circumstances presented so far, you don't have write access to any zval containers designating function parameters that have been passed to you. Of course, you can change any zval containers that you created within your function, but you mustn't change any zvals that refer to Zend-internal data!

We've only discussed the so-called ***_ex()** API so far. You may have noticed that the API functions we've used are called **zend_get_parameters_ex()** instead of **zend_get_parameters()**, **convert_to_long_ex()** instead of **convert_to_long()**, etc. The ***_ex()** functions form the so-called new "extended" Zend API. They give a minor speed increase over the old API, but as a tradeoff are only meant for providing read-only access.

Because Zend works internally with references, different variables may reference the same value. Write access to a zval container requires this container to contain an isolated value, meaning a value that's not referenced by any other containers. If a zval container were referenced by other containers and you changed the referenced zval, you would automatically change the contents of the other containers referencing this zval (because they'd simply point to the changed value and thus change their own value as well).

zend_get_parameters_ex() doesn't care about this situation, but simply returns a pointer to the desired zval containers, whether they consist of references or not. Its corresponding function in the traditional API, **zend_get_parameters()**, immediately checks for referenced values. If it finds a reference, it creates a new, isolated zval container; copies the referenced data into this newly allocated space; and then returns a pointer to the new, isolated value.

This action is called *zval separation* (or pval separation). Because the ***_ex()** API doesn't perform zval separation, it's considerably faster, while at the same time disabling write access.

To change parameters, however, write access is required. Zend deals with this situation in a special way: Whenever a parameter to a function is passed by reference, it performs automatic zval separation. This means that whenever you're calling a function like this in PHP, Zend will automatically ensure that $parameter is being passed as an isolated value, rendering it to a write-safe state:

```
my_function(&$parameter);
```

But this *is not* the case with regular parameters! All other parameters that are not passed by reference are in a read-only state.

This requires you to make sure that you're really working with a reference - otherwise you might produce unwanted results. To check for a parameter being passed by reference, you can use the macro `PZVAL_IS_REF`. This macro accepts a `zval*` to check if it is a reference or not. Examples are given in in Listing 9.6 and Figure 9.9 (see the CD-ROM for the full source).

Abbildung 32-9. Listing 9.6. Testing for referenced parameter passing.

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
    return;

/* check for parameter being passed by reference */
if (!PZVAL_IS_REF(*parameter)) {
{
    zend_error(E_WARNING, "Parameter wasn't passed by reference");
    RETURN_NULL();
}

/* make changes to the parameter */
ZVAL_LONG(*parameter, 10);
```

Assuring Write Safety for Other Parameters

You might run into a situation in which you need write access to a parameter that's retrieved with **zend_get_parameters_ex()** but not passed by reference. For this case, you can use the macro `SEPARATE_ZVAL`, which does a zval separation on the provided container. The newly generated zval is detached from internal data and has only a local scope, meaning that it can be changed or destroyed without implying global changes in the script context:

```
zval **parameter;

/* retrieve parameter */
zend_get_parameters_ex(1, &parameter);

/* at this stage, <parameter> still is connected */
/* to Zend's internal data buffers */

/* make <parameter> write-safe */
SEPARATE_ZVAL(parameter);

/* now we can safely modify <parameter> */
/* without implying global changes */
```

`SEPARATE_ZVAL` uses **emalloc()** to allocate the new zval container, which means that even if you don't deallocate this memory yourself, it will be destroyed automatically upon script termination. However, doing a lot of calls to this macro without freeing the resulting containers will clutter up your RAM.

Note: As you can easily work around the lack of write access in the "traditional" API (with **zend_get_parameters()** and so on), this API seems to be obsolete, and is not discussed further in this chapter.

Kapitel 33. Creating Variables

When exchanging data from your own extensions with PHP scripts, one of the most important issues is the creation of variables. This section shows you how to deal with the variable types that PHP supports.

Overview

To create new variables that can be seen "from the outside" by the executing script, you need to allocate a new zval container, fill this container with meaningful values, and then introduce it to Zend's internal symbol table. This basic process is common to all variable creations:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
ZEND_SET_SYMBOL(EG(active_symbol_table), "new_variable_name", new_variable);

/* the variable is now accessible to the script by using $new_variable_name */
```

The macro `MAKE_STD_ZVAL` allocates a new zval container using `ALLOC_ZVAL` and initializes it using `INIT_ZVAL`. As implemented in Zend at the time of this writing, *initializing* means setting the reference count to `1` and clearing the is_ref flag, but this process could be extended later - this is why it's a good idea to keep using `MAKE_STD_ZVAL` instead of only using `ALLOC_ZVAL`. If you want to optimize for speed (and you don't have to explicitly initialize the zval container here), you can use `ALLOC_ZVAL`, but this isn't recommended because it doesn't ensure data integrity.

`ZEND_SET_SYMBOL` takes care of introducing the new variable to Zend's symbol table. This macro checks whether the value already exists in the symbol table and converts the new symbol to a reference if so (with automatic deallocation of the old zval container). This is the preferred method if speed is not a crucial issue and you'd like to keep memory usage low.

Note that `ZEND_SET_SYMBOL` makes use of the Zend executor globals via the macro `EG`. By specifying `EG(active_symbol_table)`, you get access to the currently active symbol table, dealing with the active, local scope. The local scope may differ depending on whether the function was invoked from within a function.

If you need to optimize for speed and don't care about optimal memory usage, you can omit the check for an existing variable with the same value and instead force insertion into the symbol table by using **zend_hash_update()**:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
zend_hash_update(
```

```
    EG(active_symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

This is actually the standard method used in most modules.

The variables generated with the snippet above will always be of local scope, so they reside in the context in which the function has been called. To create new variables in the global scope, use the same method but refer to another symbol table:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
ZEND_SET_SYMBOL(&EG(symbol_table), "new_variable_name", new_variable);
```

The macro `ZEND_SET_SYMBOL` is now being called with a reference to the main, global symbol table by referring `EG(symbol_table)`.

Note: The active_symbol_table variable is a pointer, but symbol_table is not. This is why you have to use `EG(active_symbol_table)` and `&EG(symbol_table)` as parameters to `ZEND_SET_SYMBOL` - it requires a pointer.

Similarly, to get a more efficient version, you can hardcode the symbol table update:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
zend_hash_update(
    &EG(symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

Listing 9.10 shows a sample source that creates two variables - local_variable with a local scope and global_variable with a global scope (see Figure 9.7). The full example can be found on the CD-ROM.

Note: You can see that the global variable is actually not accessible from within the function. This is because it's not imported into the local scope using `global $global_variable;` in the PHP source.

Abbildung 33-1. Listing 9.10. Creating variables with different scopes.

```
ZEND_FUNCTION(variable_creation)
{
    zval *new_var1, *new_var2;

    MAKE_STD_ZVAL(new_var1);
    MAKE_STD_ZVAL(new_var2);

    ZVAL_LONG(new_var1, 10);
    ZVAL_LONG(new_var2, 5);

    ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
    ZEND_SET_SYMBOL(&EG(symbol_table), "global_variable", new_var2);

    RETURN_NULL();

}
```

Longs (Integers)

Now let's get to the assignment of data to variables, starting with longs. Longs are PHP's integers and are very simple to store. Looking at the zval.value container structure discussed earlier in this chapter, you can see that the long data type is directly contained in the union, namely in the lval field. The corresponding type value for longs is `IS_LONG` (see Listing 9.11).

Abbildung 33-2. Listing 9.11. Creation of a long.

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long->type = IS_LONG;
new_long->value.lval = 10;
```

Alternatively, you can use the macro `ZVAL_LONG`:

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Doubles (Floats)

Doubles are PHP's floats and are as easy to assign as longs, because their value is also contained directly in the union. The member in the zval.value container is dval; the corresponding type is `IS_DOUBLE`.

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double->type = IS_DOUBLE;
new_double->value.dval = 3.45;
```

Alternatively, you can use the macro `ZVAL_DOUBLE`:

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

Strings

Strings need slightly more effort. As mentioned earlier, all strings that will be associated with Zend's internal data structures need to be allocated using Zend's own memory-management functions. Referencing of static strings or strings allocated with standard routines is not allowed. To assign strings, you have to access the structure str in the zval.value container. The corresponding type is `IS_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string->type = IS_STRING;
new_string->value.str.len = strlen(string_contents);
new_string->value.str.val = estrdup(string_contents);
```

Note the usage of Zend's **estrdup()** here. Of course, you can also use the predefined macro `ZVAL_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

`ZVAL_STRING` accepts a third parameter that indicates whether the supplied string contents should be duplicated (using **estrdup()**). Setting this parameter to `1` causes the string to be duplicated; `0` simply uses the supplied pointer for the variable contents. This is most useful if you want to create a new variable referring to a string that's already allocated in Zend internal memory.

If you want to truncate the string at a certain position or you already know its length, you can use `ZVAL_STRINGL(zval, string, length, duplicate)`, which accepts an explicit string length to be set for the new string. This macro is faster than `ZVAL_STRING` and also binary-safe.

To create empty strings, set the string length to `0` and use `empty_string` as contents:

```
new_string->type = IS_STRING;
new_string->value.str.len = 0;
new_string->value.str.val = empty_string;
```

Of course, there's a macro for this as well (`ZVAL_EMPTY_STRING`):

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

Booleans

Booleans are created just like longs, but have the type `IS_BOOL`. Allowed values in lval are `0` and `1`:

```
zval *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool->type = IS_BOOL;
new_bool->value.lval = 1;
```

The corresponding macros for this type are `ZVAL_BOOL` (allowing specification of the value) as well as `ZVAL_TRUE` and `ZVAL_FALSE` (which explicitly set the value to `TRUE` and `FALSE`, respectively).

Arrays

Arrays are stored using Zend's internal hash tables, which can be accessed using the **zend_hash_*()** API. For every array that you want to create, you need a new hash table handle, which will be stored in the ht member of the zval.value container.

There's a whole API solely for the creation of arrays, which is extremely handy. To start a new array, you call **array_init()**.

```
zval *new_array;

MAKE_STD_ZVAL(new_array);

if(array_init(new_array) != SUCCESS)
{
    // do error handling here
}
```

If **array_init()** fails to create a new array, it returns `FAILURE`.

To add new elements to the array, you can use numerous functions, depending on what you want to do. Tables 9.8, 9.9, and 9.10 describe these functions. All functions return `FAILURE` on failure and `SUCCESS` on success.

Abbildung 33-3. Table 9.8. Zend's API for Associative Arrays

```
Note: The functions in Table 9.8 all operate on the array "array" with the key "key".
The key string doesn't have to reside in Zend internal memory; it will be du-
plicated by the API.
```

Function	Description
add_assoc_long(zval *array, char *key, long n);()	Adds an element of type `long`.
add_assoc_unset(zval *array, char *key);()	Adds an unset element.
add_assoc_bool(zval *array, char *key, int b);()	Adds a Boolean element.
add_assoc_resource(zval *array, char *key, int r);()	Adds a resource to the array.
add_assoc_double(zval *array, char *key, double d);()	Adds a floating-point value.
add_assoc_string(zval *array, char *key, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_assoc_stringl(zval *array, char *key, char *str, uint length, int duplicate); ()	Adds a string with the desired length length to the array. Otherwise, behaves like **add_assoc_string()**.

Abbildung 33-4. Table 9.9. Zend's API for Indexed Arrays, Part 1

```
Note: The functions in Table 9.9 all operate on the array "array" with the in-
dex "idx".
The index is always an integer.
```

Function	Description

add_index_long(zval *array, uint idx, long n);()	Adds an element of type `long`.
add_index_unset(zval *array, uint idx);()	Adds an unset element.
add_index_bool(zval *array, uint idx, int b);()	Adds a Boolean element.
add_index_resource(zval *array, uint idx, int r);()	Adds a resource to the array.
add_index_double(zval *array, uint idx, double d);()	Adds a floating-point value.
add_index_string(zval *array, uint idx, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

Abbildung 33-5. Table 9.10. Zend's API for Indexed Arrays, Part 2

```
Note: The functions in Table 9.10 all operate on the array "array".
These functions automatically generate a new index based on the highest in-
dex found in the array.
```

Function	Description
add_next_index_long(zval *array, long n);()	Adds an element of type `long`.
add_next_index_unset(zval *array);()	Adds an unset element.
add_next_index_bool(zval *array, int b);()	Adds a Boolean element.
add_next_index_resource(zval *array, int r);()	Adds a resource to the array.
add_next_index_double(zval *array, double d);()	Adds a floating-point value.
add_next_index_string(zval *array, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

All these functions provide a handy abstraction to Zend's internal hash API. Of course, you can also use the hash functions directly - for example, if you already have a zval container allocated that you want to insert into an array. This is done using **zend_hash_update()()** for associative arrays (see Listing 9.12) and **zend_hash_index_update()** for indexed arrays (see Listing 9.13):

Abbildung 33-6. Listing 9.12. Adding an element to an associative array.

```
zval *new_array, *new_element;
char *key = "element_key";
```

```
MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_update(new_array->value.ht, key, strlen(key) + 1, (void *)&new_element, si-
zeof(zval *), NULL) == FAILURE)
{
    // do error handling here
}
```

Abbildung 33-7. Listing 9.13. Adding an element to an indexed array.

```
zval *new_array, *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_index_update(new_array->value.ht, key, (void *)&new_element, si-
zeof(zval *), NULL) == FAILURE)
{
    // do error handling here
}
```

To emulate the functionality of **add_next_index_*()**, you can use this:

```
zend_hash_next_index_insert(ht, zval **new_element, sizeof(zval *), NULL)
```

Note: To return arrays from a function, use **array_init()** and all following actions on the predefined variable return_value (given as argument to your exported function; see the earlier discussion of the call interface). You do not have to use `MAKE_STD_ZVAL` on this.

Tip: To avoid having to write `new_array->value.ht` every time, you can use `HASH_OF(new_array)`, which is also recommended for compatibility and style reasons.

Objects

Since objects can be converted to arrays (and vice versa), you might have already guessed that they have a lot of similarities to arrays in PHP. Objects are maintained with the same hash functions, but there's a different API for creating them.

To initialize an object, you use the function **object_init()**:

```
zval *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object) != SUCCESS)
{
    // do error handling here
}
```

You can use the functions described in Table 9.11 to add members to your object.

Abbildung 33-8. Table 9.11. Zend's API for Object Creation

```
     Note: All functions in Table 9.11 work on the object "object" with the key "key". Th
ber name,
     so the resulting member can be accessed via $object->key.
```

Function	Description
add_property_long(zval *object, char *key, long l);()	Adds a long to the object.
add_property_unset(zval *object, char *key);()	Adds an unset property to the
add_property_bool(zval *object, char *key, int b);()	Adds a Boolean to the object
add_property_resource(zval *object, char *key, long r);()	Adds a resource to the object
add_property_double(zval *object, char *key, double d);()	Adds a double to the object.
add_property_string(zval *object, char *key, char *str, int duplicate);()	Adds a string to the object.
add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	Adds a string of the specified
add_property_zval(zval *obect, char *key, zval *container):()	Adds a `zval` container to the

Resources

Resources are a special kind of data type in PHP. The term *resources* doesn't really refer to any special kind of data, but to an abstraction method for maintaining any kind of information. Resources are kept in a special resource list within Zend. Each entry in the list has a correspondending type definition that denotes the kind of resource to which it refers. Zend then internally manages all references to this resource. Access to a resource is never possible directly - only via a provided API. As soon as all references to a specific resource are lost, a corresponding shutdown function is called.

For example, resources are used to store database links and file descriptors. The *de facto* standard implementation can be found in the MySQL module, but other modules such as the Oracle module also make use of resources.

Anmerkung: In fact, a resource can be a pointer to anything you need to handle in your functions (e.g. pointer to a structure) and the user only has to pass a single resource variable to your function.

To create a new resource you need to register a resource destruction handler for it. Since you can store any kind of data as a resource, Zend needs to know how to free this resource if its not longer needed. This works by registering your own resource destruction handler to Zend which in turn gets called by Zend whenever your resource can be freed (whether manually or automatically). Registering your resource handler within Zend returns you the **resource type handle** for that resource. This handle is needed whenever you want to access a resource of this type later and is most of time stored in a global static variable within your extension. There is no need to worry about thread safety here because you only register your resource handler once during module initialization.

The Zend function to register your resource handler is defined as:

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pld,
pe_name, int module_number);
```

There are two different kinds of resource destruction handlers you can pass to this function: a handler for normal resources and a handler for persistent resources. Persistent resources are for example used for database connection. When registering a resource, either of these handlers must be given. For the other handler just pass `NULL`.

zend_register_list_destructors_ex() accepts the following parameters:

`ld`	Normal resource destruction handler callback
`pld`	Pesistent resource destruction handler callback
`type_name`	A string specifying the name of your resource. It's always a good thing to specify an unique name wi
`module_number`	The `module_number` is automatically available in your `PHP_MINIT_FUNCTION` function and therefo

The return value is an unique integer ID for your **resource type**.

The resource destruction handler (either normal or persistent resources) has the following prototype:

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

The passed `rsrc` is a pointer to the following structure:

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;

} zend_rsrc_list_entry;
```

The member `void *ptr` is the actual pointer to your resource.

Now we know how to start things, we define our own resource we want register within Zend. It is only a simple structure with two integer members:

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

Our resource destruction handler is probably going to look something like this:

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // You most likely cast the void pointer to your structure type

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // Now do whatever needs to be done with you resource. Closing
    // Files, Sockets, freeing additional memory, etc.
    // Also, don't forget to actually free the memory for your resource too!

    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

Anmerkung: One important thing to mention: If your resource is a rather complex structure which also contains pointers to memory you allocated during runtime you have to free them **before** freeing the resource itself!

Now that we have defined

1. what our resource is and
2. our resource destruction handler

we can go on and do the rest of the steps:

1. create a global variable within the extension holding the resource ID so it can be accessed from every function which needs it
2. define the resource name
3. write the resource destruction handler
4. and finally register the handler

```
// Somewhere in your extension, define the variable for your registered resources.
// If you wondered what 'le' stands for: it simply means 'list entry'.
static int le_myresource;

// It's nice to define your resource name somewhere
#define le_myresource_name  "My type of resource"

[...]

// Now actually define our resource destruction handler
```

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;
    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}

[...]

PHP_MINIT_FUNCTION(my_extension) {

    // Note that 'module_number' is already provided through the
    // PHP_MINIT_FUNCTION() function definition.

    le_myresource = zend_register_resource_destructors_ex(my_destruction_handler, NUL

    // You can register additional resources, initialize
    // your global vars, constants, whatever.
}
```

To actually register a new resource you use can either use the **zend_register_resource()** function or the **ZEND_REGISTER_RESOURE()** macro, both defined in zend_list.h . Although the arguments for both map 1:1 it's a good idea to always use macros to be upwards compatible:

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

`rsrc_result`	This is an already initialized `zval *` container.
`rsrc_pointer`	Your resource pointer you want to store.
`rsrc_type`	The type which you received when you registered the resource destruction handler. If you followed the

The return value is an unique integer identifier for that resource.

What is really going on when you register a new resource is it gets inserted in an internal list in Zend and the result is just stored in the given `zval *` container:

```
rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);

if (rsrc_result) {
    rsrc_result->value.lval = rsrc_id;
    rsrc_result->type = IS_RESOURCE;
}

return rsrc_id;
```

The returned `rsrc_id` uniquly identifies the newly registered resource. You can use the macro `RETURN_RESOURE` to return it to the user:

```
RETURN_RESOURCE(rsrc_id)
```

Anmerkung: It is common practice that if you want to return the resource immidiately to the user you specify the `return_value` as the `zval *` container.

Zend now keeps track of all references to this resource. As soon as all references to the resource are lost, the destructor that you previously registered for this resource is called. The nice thing about this setup is that you don't have to worry about memory leakages introduced by allocations in your module - just register all memory allocations that your calling script will refer to as resources. As soon as the script decides it doesn't need them anymore, Zend will find out and tell you.

Now that the user got his resource, at some point he is passing it back to one of your functions. The value.lval inside the `zval *` container contains the key to your resource and thus can be used to fetch the resource with the following macro: `ZEND_FETCH_RESOURCE`:

```
ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resour
```

`rsrc`	This is your pointer which will point to your previously registered resource.
`rsrc_type`	This is the typecast argument for your pointer, e.g. `myresource *`.
`rsrc_id`	This is the address of the `zval *`container the user passed to your function, e.g. `&z_resourc`
`default_rsrc_id`	This integer specifies the default resource `ID` if no resource could be fetched or -1.
`resource_type_name`	This is the name of the requested resource. It's a string and is used when the resource can't be
`resource_type`	The `resource_type` you got back when registering the resource destruction handler. In our e

This macro has no return value. It is for the developers convenience and takes care of TSRMLS arguments passing and also does check if the resource could be fetched. It throws a warning message and returns the current PHP function with `NULL` if there was a problem retrieving the resource.

To force removal of a resource from the list, use the function **zend_list_delete()**. You can also force the reference count to increase if you know that you're creating another reference for a previously allocated value (for example, if you're automatically reusing a default database link). For this case, use the function **zend_list_addref()**. To search for previously allocated resource entries, use **zend_list_find()**. The complete API can be found in `zend_list.h`.

Macros for Automatic Global Variable Creation

In addition to the macros discussed earlier, a few macros allow easy creation of simple global variables. These are nice to know in case you want to introduce global flags, for example. This is somewhat bad practice, but Table 9.12 describes macros that do exactly this task. They don't need any zval allocation; you simply have to supply a variable name and value.

Abbildung 33-9. Table 9.12. Macros for Global Variable Creation

```
Note: All macros in Table 9.12 create a global variable of the name "name" with the
```

Macro	Description
`SET_VAR_STRING(name, value)`	Creates a new string.
`SET_VAR_STRINGL(name, value, length)`	Creates a new string of the specified length. This macro is faster than `SET_VAR_STRING` and also binary-safe.
`SET_VAR_LONG(name, value)`	Creates a new long.

SET_VAR_DOUBLE(name, value)	Creates a new double.

Creating Constants

Zend supports the creation of true constants (as opposed to regular variables). Constants are accessed without the typical dollar sign ($) prefix and are available in all scopes. Examples include TRUE and FALSE, to name just two.

To create your own constants, you can use the macros in Table 9.13. All the macros create a constant with the specified name and value.

You can also specify flags for each constant:

- CONST_CS - This constant's name is to be treated as case sensitive.
- CONST_PERSISTENT - This constant is persistent and won't be "forgotten" when the current process carrying this constant shuts down.

To use the flags, combine them using a inary OR:

```
// register a new constant of type "long"
    REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
    CONST_PERSISTENT);
```

There are two types of macros - REGISTER_*_CONSTANT andREGISTER_MAIN_*_CONSTANT. The first type creates constants bound to the current module. These constants are dumped from the symbol table as soon as the module that registered the constant is unloaded from memory. The second type creates constants that remain in the symbol table independently of the module.

Abbildung 33-10. Table 9.13. Macros for Creating Constants

Macro	
REGISTER_LONG_CONSTANT(name, value, flags)	REGISTER_MAIN_LONG_CONSTANT(name, value, flags
REGISTER_DOUBLE_CONSTANT(name, value, flags)	REGISTER_MAIN_DOUBLE_CONSTANT(name, value, f
REGISTER_STRING_CONSTANT(name, value, flags)	REGISTER_MAIN_STRING_CONSTANT(name, value, f
REGISTER_STRINGL_CONSTANT(name, value, length, flags)	REGISTER_MAIN_STRINGL_CONSTANT(name

Kapitel 34. Duplicating Variable Contents: The Copy Constructor

Sooner or later, you may need to assign the contents of one zval container to another. This is easier said than done, since the zval container doesn't contain only type information, but also references to places in Zend's internal data. For example, depending on their size, arrays and objects may be nested with lots of hash table entries. By assigning one zval to another, you avoid duplicating the hash table entries, using only a reference to them (at most).

To copy this complex kind of data, use the *copy constructor*. Copy constructors are typically defined in languages that support operator overloading, with the express purpose of copying complex types. If you define an object in such a language, you have the possibility of overloading the "=" operator, which is usually responsible for assigning the contents of the lvalue (result of the evaluation of the left side of the operator) to the rvalue (same for the right side).

Overloading means assigning a different meaning to this operator, and is usually used to assign a function call to an operator. Whenever this operator would be used on such an object in a program, this function would be called with the lvalue and rvalue as parameters. Equipped with that information, it can perform the operation it intends the "=" operator to have (usually an extended form of copying).

This same form of "extended copying" is also necessary for PHP's zval containers. Again, in the case of an array, this extended copying would imply re-creation of all hash table entries relating to this array. For strings, proper memory allocation would have to be assured, and so on.

Zend ships with such a function, called **zend_copy_ctor()** (the previous PHP equivalent was **pval_copy_constructor()**).

A most useful demonstration is a function that accepts a complex type as argument, modifies it, and then returns the argument:

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
   return;
}

// do modifications to the parameter here

// now we want to return the modified container:
*return_value == *parameter;
zval_copy_ctor(return_value);
```

The first part of the function is plain-vanilla argument retrieval. After the (left out) modifications, however, it gets interesting: The container of parameter is assigned to the (predefined) return_value container. Now, in order to effectively duplicate its contents, the copy constructor is called. The copy constructor works directly with the supplied argument, and the standard return values are `FAILURE` on failure and `SUCCESS` on success.

If you omit the call to the copy constructor in this example, both parameter and return_value would point to the same internal data, meaning that return_value would be an illegal additional reference to the same data structures. Whenever changes occurred in the data that parameter points to, return_value might be affected. Thus, in order to create separate copies, the copy constructor must be used.

The copy constructor's counterpart in the Zend API, the destructor **zval_dtor()**, does the opposite of the constructor.

Kapitel 35. Returning Values

Returning values from your functions to PHP was described briefly in an earlier section; this section gives the details. Return values are passed via the return_value variable, which is passed to your functions as argument. The return_value argument consists of a zval container (see the earlier discussion of the call interface) that you can freely modify. The container itself is already allocated, so you don't have to run `MAKE_STD_ZVAL` on it. Instead, you can access its members directly.

To make returning values from functions easier and to prevent hassles with accessing the internal structures of the zval container, a set of predefined macros is available (as usual). These macros automatically set the correspondent type and value, as described in Tables 9.14 and 9.15.

Abbildung 35-1. Table 9.14. Predefined Macros for Returning Values from a Function

```
Note: The macros in Table 9.14 automatically return
    from your function.
```

Macro	Description
`RETURN_RESOURCE(resource)`	Returns a resource.
`RETURN_BOOL(bool)`	Returns a Boolean.
`RETURN_NULL()`	Returns nothing (a NULL value).
`RETURN_LONG(long)`	Returns a long.
`RETURN_DOUBLE(double)`	Returns a double.
`RETURN_STRING(string, duplicate)`	Returns a string. The duplicate flag indicates whether the string should be duplicated using **estrdup()**.
`RETURN_STRINGL(string, length, duplicate)`	Returns a string of the specified length; otherwise, behaves like `RETURN_STRING`. This macro is faster and binary-safe, however.
`RETURN_EMPTY_STRING()`	Returns an empty string.
`RETURN_FALSE`	Returns Boolean false.
`RETURN_TRUE`	Returns Boolean true.

Abbildung 35-2. Table 9.15. Predefined Macros for Setting the Return Value of a Function

```
Note: The macros in Table 9.15 only set the
return value; they don't return from your function.
```

Macro	Description
`RETVAL_RESOURCE(resource)`	Sets the return value to the specified resource.
`RETVAL_BOOL(bool)`	Sets the return value to the specified Boolean value.
`RETVAL_NULL`	Sets the return value to NULL.
`RETVAL_LONG(long)`	Sets the return value to the specified long.
`RETVAL_DOUBLE(double)`	Sets the return value to the specified double.
`RETVAL_STRING(string, duplicate)`	Sets the return value to the specified string and duplicates it to Zend internal memory if desired (see also `RETURN_STRING`).

`RETVAL_STRINGL(string, length, duplicate)`	Sets the return value to the specified string and forces the length to become length (see also `RETVAL_STRING`). This macro is faster and binary-safe, and should be used whenever the string length is known.
`RETVAL_EMPTY_STRING`	Sets the return value to an empty string.
`RETVAL_FALSE`	Sets the return value to Boolean false.
`RETVAL_TRUE`	Sets the return value to Boolean true.

Complex types such as arrays and objects can be returned by using **array_init()** and **object_init()**, as well as the corresponding hash functions on return_value. Since these types cannot be constructed of trivial information, there are no predefined macros for them.

Kapitel 36. Printing Information

Often it's necessary to print messages to the output stream from your module, just as print() would be used within a script. PHP offers functions for most generic tasks, such as printing warning messages, generating output for phpinfo(), and so on. The following sections provide more details. Examples of these functions can be found on the CD-ROM.

zend_printf()

zend_printf() works like the standard printf(), except that it prints to Zend's output stream.

zend_error()

zend_error() can be used to generate error messages. This function accepts two arguments; the first is the error type (see `zend_errors.h`), and the second is the error message.

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

Table 9.16 shows a list of possible values (see Figure 9.8). These values are also referred to in `php.ini`. Depending on which error type you choose, your messages will be logged.

Abbildung 36-1. Table 9.16. Zend's Predefined Error Messages.

Error	Description
`E_ERROR`	Signals an error and terminates execution of the script immediately .
`E_WARNING`	Signals a generic warning. Execution continues.
`E_PARSE`	Signals a parser error. Execution continues.
`E_NOTICE`	Signals a notice. Execution continues. Note that by default the display of this type of error mes
`E_CORE_ERROR`	Internal error by the core; shouldn't be used by user-written modules.
`E_COMPILE_ERROR`	Internal error by the compiler; shouldn't be used by user-written modules.
`E_COMPILE_WARNING`	Internal warning by the compiler; shouldn't be used by user-written modules.

Abbildung 36-2. Figure 9.8. Display of warning messages in the browser.

Including Output in phpinfo()

After creating a real module, you'll want to show information about the module in phpinfo() (in addition to the module name, which appears in the module list by default). PHP allows you to create your own section in the phpinfo() output with the `ZEND_MINFO()` function. This function should be placed in the module descriptor block (discussed earlier) and is always called whenever a script calls phpinfo().

PHP automatically prints a section in phpinfo() for you if you specify the `ZEND_MINFO` function, including the module name in the heading. Everything else must be formatted and printed by you.

Typically, you can print an HTML table header using **php_info_print_table_start()** and then use the standard functions **php_info_print_table_header()** and **php_info_print_table_row()**. As arguments, both take the number of columns (as integers) and the column contents (as strings). Listing 9.14 shows a source example; Figure 9.9 shows the output. To print the table footer, use **php_info_print_table_end()**.

Abbildung 36-3. Listing 9.14. Source code and screenshot for output in phpinfo().

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

Execution Information

You can also print execution information, such as the current file being executed. The name of the function currently being executed can be retrieved using the function **get_active_function_name()**. This function returns a pointer to the function name and doesn't accept any arguments. To retrieve the name of the file currently being executed, use **zend_get_executed_filename()**. This function accesses the executor globals, which are passed to it using the `TSRMLS_C` macro. The executor globals are automatically available to every function that's called directly by Zend (they're part of the `INTERNAL_FUNCTION_PARAMETERS` described earlier in this chapter). If you want to access the executor globals in another function that doesn't have them available automatically, call the macro `TSRMLS_FETCH()` once in that function; this will introduce them to your local scope.

Finally, the line number currently being executed can be retrieved using the function **zend_get_executed_lineno()**. This function also requires the executor globals as arguments. For examples of these functions, see Listing 9.15 and Figure 9.10.

Abbildung 36-4. Listing 9.15. Printing execution information.

```
zend_printf("The name of the current function is %s<br>", get_active_function_name(TSRMLS
zend_printf("The file currently executed is %s<br>", zend_get_executed_filename(TSRMLS_C)
zend_printf("The current line being executed is %i<br>", zend_get_executed_lineno(TSRMLS_
```

Kapitel 37. Startup and Shutdown Functions

Startup and shutdown functions can be used for one-time initialization and deinitialization of your modules. As discussed earlier in this chapter (see the description of the Zend module descriptor block), there are global, module, and request startup and shutdown events.

The global startup functions are called once when PHP starts up; similarly, the global shutdown functions are called once when PHP shuts down. Please note that they’re really only called *once*, not when a new Apache process is being created!

The module startup and shutdown functions are called whenever a module is loaded and needs initialization; the request startup and shutdown functions are called every time a request is processed (meaning that a file is being executed).

For dynamic extensions, module and request startup/shutdown events happen at the same time.

Declaration and implementation of these functions can be done with macros; see the earlier section "Declaration of the Zend Module Block" for details.

Kapitel 38. Calling User Functions

You can call user functions from your own modules, which is very handy when implementing callbacks; for example, for array walking, searching, or simply for event-based programs.

User functions can be called with the function **call_user_function_ex()**. It requires a hash value for the function table you want to access, a pointer to an object (if you want to call a method), the function name, return value, number of arguments, argument array, and a flag indicating whether you want to perform zval separation.

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

Note that you don't have to specify both function_table and object; either will do. If you want to call a method, you have to supply the object that contains this method, in which case **call_user_function()**automatically sets the function table to this object's function table. Otherwise, you only need to specify function_table and can set object to `NULL`.

Usually, the default function table is the "root" function table containing all function entries. This function table is part of the compiler globals and can be accessed using the macro `CG`. To introduce the compiler globals to your function, call the macro `TSRMLS_FETCH` once.

The function name is specified in a zval container. This might be a bit surprising at first, but is quite a logical step, since most of the time you'll accept function names as parameters from calling functions within your script, which in turn are contained in zval containers again. Thus, you only have to pass your arguments through to this function. This zval must be of type `IS_STRING`.

The next argument consists of a pointer to the return value. You don't have to allocate memory for this container; the function will do so by itself. However, you have to destroy this container (using **zval_dtor()**) afterward!

Next is the parameter count as integer and an array containing all necessary parameters. The last argument specifies whether the function should perform zval separation - this should always be set to `0`. If set to `1`, the function consumes less memory but fails if any of the parameters need separation.

Listing 9.16 and Figure 9.11 show a small demonstration of calling a user function. The code calls a function that's supplied to it as argument and directly passes this function's return value through as its own return value. Note the use of the constructor and destructor calls at the end - it might not be necessary to do it this way here (since they should be separate values, the assignment might be safe), but this is bulletproof.

Abbildung 38-1. Listing 9.16. Calling user functions.

```
zval **function_name;
zval *retval;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &function_name) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

if((*function_name)->type != IS_STRING)
{
    zend_error(E_ERROR, "Function requires string argument");
}

TSRMSLS_FETCH();
```

```
if(call_user_function_ex(CG(function_table), NULL, *function_name, &retval, 0, NULL, 0)
{
    zend_error(E_ERROR, "Function call failed");
}

zend_printf("We have %i as type<br>", retval->type);

*return_value = *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);
```

```
<?php

dl("call_userland.so");

function test_function()
{

    print("We are in the test function!<br>");

    return("hello");

}

$return_value = call_userland("test_function");

print("Return value: \"$return_value\"<br>");
?>
```

Kapitel 39. Initialization File Support

PHP 4 features a redesigned initialization file support. It's now possible to specify default initialization entries directly in your code, read and change these values at runtime, and create message handlers for change notifications.

To create an .ini section in your own module, use the macros `PHP_INI_BEGIN()` to mark the beginning of such a section and `PHP_INI_END()` to mark its end. In between you can use `PHP_INI_ENTRY()` to create entries.

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",  "has_string_value", PHP_INI_ALL, NULL)
PHP_INI_ENTRY("second_ini_entry", "2",                PHP_INI_SYSTEM, OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",  "xyz",              PHP_INI_USER, NULL)
PHP_INI_END()
```

The `PHP_INI_ENTRY()` macro accepts four parameters: the entry name, the entry value, its change permissions, and a pointer to a change-notification handler. Both entry name and value must be specified as strings, regardless of whether they really are strings or integers.

The permissions are grouped into three sections:`PHP_INI_SYSTEM` allows a change only directly in the `php3.ini` file; `PHP_INI_USER` allows a change to be overridden by a user at runtime using additional configuration files, such as `.htaccess`; and `PHP_INI_ALL` allows changes to be made without restrictions. There's also a fourth level, `PHP_INI_PERDIR`, for which we couldn't verify its behavior yet.

The fourth parameter consists of a pointer to a change-notification handler. Whenever one of these initialization entries is changed, this handler is called. Such a handler can be declared using the `PHP_INI_MH` macro:

```
PHP_INI_MH(OnChangeSecond);             // handler for ini-entry "second_ini_entry"

// specify ini-entries here

PHP_INI_MH(OnChangeSecond)
{

    zend_printf("Message caught, our ini entry has been changed to %s<br>", new_value);

    return(SUCCESS);

}
```

The new value is given to the change handler as string in the variable new_value. When looking at the definition of `PHP_INI_MH`, you actually have a few parameters to use:

```
#define PHP_INI_MH(name) int name(php_ini_entry *entry, char *new_value,
                                  uint new_value_length, void *mh_arg1,
                                  void *mh_arg2, void *mh_arg3)
```

All these definitions can be found in `php_ini.h`. Your message handler will have access to a structure that contains the full entry, the new value, its length, and three optional arguments. These optional arguments can be specified with the additional macros `PHP_INI_ENTRY1` (allowing one additional argument), `PHP_INI_ENTRY2` (allowing two additional arguments), and `PHP_INI_ENTRY3` (allowing three additional arguments).

The change-notification handlers should be used to cache initialization entries locally for faster access or to perform certain tasks that are required if a value changes. For example, if a constant

connection to a certain host is required by a module and someone changes the hostname, automatically terminate the old connection and attempt a new one.

Access to initialization entries can also be handled with the macros shown in Table 9.17.

Abbildung 39-1. Table 9.17. Macros to Access Initialization Entries in PHP

Macro	Description
`INI_INT(name)`	Returns the current value of entry `name` as integer (long).
`INI_FLT(name)`	Returns the current value of entry `name` as float (double).
`INI_STR(name)`	Returns the current value of entry `name` as string. *Note:* This string is not duplicated, but inst
`INI_BOOL(name)`	Returns the current value of entry `name` as Boolean (defined as zend_bool, which currently m
`INI_ORIG_INT(name)`	Returns the original value of entry `name` as integer (long).
`INI_ORIG_FLT(name)`	Returns the original value of entry `name` as float (double).
`INI_ORIG_STR(name)`	Returns the original value of entry `name` as string. Note: This string is not duplicated, but inst
`INI_ORIG_BOOL(name)`	Returns the original value of entry `name` as Boolean (defined as zend_bool, which currently r

Finally, you have to introduce your initialization entries to PHP. This can be done in the module startup and shutdown functions, using the macros `REGISTER_INI_ENTRIES()` and `UNREGISTER_INI_ENTRIES()`:

```
ZEND_MINIT_FUNCTION(mymodule)
{

    REGISTER_INI_ENTRIES();

}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{

    UNREGISTER_INI_ENTRIES();

}
```

Kapitel 40. Where to Go from Here

You’ve learned a lot about PHP. You now know how to create dynamic loadable modules and statically linked extensions. You’ve learned how PHP and Zend deal with internal storage of variables and how you can create and access these variables. You know quite a set of tool functions that do a lot of routine tasks such as printing informational texts, automatically introducing variables to the symbol table, and so on.

Even though this chapter often had a mostly "referential" character, we hope that it gave you insight on how to start writing your own extensions. For the sake of space, we had to leave out a lot; we suggest that you take the time to study the header files and some modules (especially the ones in the `ext/standard` directory and the MySQL module, as these implement commonly known functionality). This will give you an idea of how other people have used the API functions - particularly those that didn’t make it into this chapter.

Kapitel 41. Reference: Some Configuration Macros

config.m4

The file `config.m4` is processed by `buildconf` and must contain all the instructions to be executed during configuration. For example, these can include tests for required external files, such as header files, libraries, and so on. PHP defines a set of macros that can be used in this process, the most useful of which are described in Table 9.18.

Abbildung 41-1. Table 9.18. M4 Macros for `config.m4`

Macro	Description
`AC_MSG_CHECKING(message)`	Prints a "checking <mess
`AC_MSG_RESULT(value)`	Gives the result to `AC_MS`
`AC_MSG_ERROR(message)`	Prints message as error m
`AC_DEFINE(name,value,description)`	Adds `#define` to `php_c`
`AC_ADD_INCLUDE(path)`	Adds a compiler include
`AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)`	Specifies an additional lib
`AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)`	Quite a powerful macro,
`PHP_EXTENSION(modulename, [shared])`	This macro is a *must* to c

Kapitel 42. API Macros

A set of macros was introduced into Zend's API that simplify access to zval containers (see Table 9.19).

Abbildung 42-1. Table 9.19. API Macros for Accessing zval Containers

Macro	Refers to
`Z_LVAL(zval)`	(zval).value.lval
`Z_DVAL(zval)`	(zval).value.dval
`Z_STRVAL(zval)`	(zval).value.str.val
`Z_STRLEN(zval)`	(zval).value.str.len
`Z_ARRVAL(zval)`	(zval).value.ht
`Z_LVAL_P(zval)`	(*zval).value.lval
`Z_DVAL_P(zval)`	(*zval).value.dval
`Z_STRVAL_P(zval_p)`	(*zval).value.str.val
`Z_STRLEN_P(zval_p)`	(*zval).value.str.len
`Z_ARRVAL_P(zval_p)`	(*zval).value.ht
`Z_LVAL_PP(zval_pp)`	(**zval).value.lval
`Z_DVAL_PP(zval_pp)`	(**zval).value.dval
`Z_STRVAL_PP(zval_pp)`	(**zval).value.str.val
`Z_STRLEN_PP(zval_pp)`	(**zval).value.str.len
`Z_ARRVAL_PP(zval_pp)`	(**zval).value.ht

Teil VI. FAQ: Frequently Asked Questions

Kapitel 43. General Information

This section holds the most general questions about PHP: what it is and what it does.

1. What is PHP?

From the preface of the manual:

PHP is an HTML-embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly.

A nice introduction to PHP by Stig Sæther Bakken can be found here (http://www.zend.com/zend/art/intro.php) on the Zend website. Also, much of the PHP Conference Material (http://conf.php.net/) is freely available.

2. What does PHP stand for?

PHP stands for *PHP: Hypertext Preprocessor*. This confuses many people because the first word of the acronym is the acronym. This type of acronym is called a recursive acronym. The curious can visit Free On-Line Dictionary of Computing (http://www.foldoc.org/) for more information on recursive acronyms.

3. What is the relation between the versions?

PHP/FI 2.0 is an early and no longer supported version of PHP. PHP 3 is the successor to PHP/FI 2.0 and is a lot nicer. PHP 4 is the latest generation of PHP, which uses the Zend engine (http://www.zend.com/) under the hood.

4. Can I run several versions of PHP at the same time?

Yes. See the `INSTALL` file that is included in the PHP 4 source distribution. Also, read the related appendix.

5. What are the differences between PHP 3 and PHP 4?

There are a couple of articles (http://www.zend.com/zend/art/) written on this by the authors of PHP 4. Here's a list of some of the more important new features:

- Extended API module
- Generalized build process under UNIX
- Generic web server interface that also supports multi-threaded web servers
- Improved syntax highlighter
- Native HTTP session support
- Output buffering support
- More powerful configuration system
- Reference counting

Please see the What's new in PHP 4 overview (http://www.zend.com/zend/whats-new.php) for a detailed explanation of these features and more. If you're migrating from PHP 3 to PHP 4, also read the related appendix.

6. I think I found a bug! Who should I tell?

You should go to the PHP Bug Database and make sure the bug isn't a known bug. If you don't see it in the database, use the reporting form to report the bug. It is important to use the bug database instead of just sending an email to one of the mailing lists because the bug will have a tracking

number assigned and it will then be possible for you to go back later and check on the status of the bug. The bug database can be found at http://bugs.php.net/.

Kapitel 44. Mailing lists

This section holds questions about how to get in touch with the PHP community. The best way is the mailing lists.

1. Are there any PHP mailing lists?

Of course! There are many mailing lists for several subjects. A whole list of mailing lists can be found on our Support (http://www.php.net/support.php) page.

The most general mailing list is `php-general`. To subscribe, send mail to php-general-subscribe@lists.php.net (mailto:php-general-subscribe@lists.php.net). You don't need to include anything special in the subject or body of the message. To unsubscribe, send mail to php-general-unsubscribe@lists.php.net (mailto:php-general-unsubscribe@lists.php.net).

You can also subscribe and unsubscribe using the web interface on our Support (http://www.php.net/support.php) page.

2. Are there any other communities?

There are countless of them around the world. We have links for example to some IRC servers and foreign language mailing lists on our Support (http://www.php.net/support.php) page.

3. Help! I can't seem to subscribe/unsubscribe to/from one of the mailing lists!

If you have problems subscribing to or unsubscribing from the php-general mailing list, it may be because the mailing list software can't figure out the correct mailing address to use. If your email address was `joeblow@example.com`, you can send your subscription request to `php-general-subscribe-joeblow=example.com@lists.php.net`, or your unsubscription request to `php-general-unsubscribe-joeblow=example.com@lists.php.net`. Use similar addresses for the other mailing lists.

4. Is there an archive of the mailing lists anywhere?

Yes, you will find a list of archive sites on the Support (http://www.php.net/support.php) page. The mailing list articles are also archived as news messages. You can access the news server at news://news.php.net/ with a news client. There is also an experimental web interface for the news server at http://news.php.net/

5. What can I ask the mailing list?

Since PHP is growing more and more popular by the day the traffic has increased on the php-general mailing list and as of now the list gets about 150 to 200 posts a day. Because of this it is in everyones interest that you use the list as a last resort when you have looked everywhere else.

Before you post to the list please have a look in this FAQ and the manual to see if you can find the help there. If there is nothing to be found there try out the mailing list archives (see above). If you're having problem with installing or configuring PHP please read through all included documentation and README's. If you still can't find any information that helps you out you're more than welcome to use the mailing list.

6. What information should I include when posting to the mailing list?

Posts like "I can't get PHP up and running! Help me! What is wrong?" are of absolutely no use to anyone. If you're having problems getting PHP up and running you must include what operating system you are running on, what version of PHP you're trying to set up, how you got it (pre-compiled, CVS, RPMs and so on), what you have done so far, where you got stuck and the exact error message.

This goes for any other problem as well. You have to include information on what you have done, where you got stuck, what you're trying to do and, if applicable, exact error messages. If you're having problems with your source code you need to include the part of the code that isn't working. Do not include more code than necessary though! It makes the post hard to read and a lot of people might just skip it all together because of this. If you're unsure about how much information to include in the mail it's better that you include to much than to little.

Another important thing to remember is to summarize your problem on the subject line. A subject like "HELP MEEEE!!!" or "What is the problem here?" will be ignored by the majority of the readers.

Kapitel 45. Obtaining PHP

This section has details about PHP download locations, and OS issues.

1. Where can I obtain PHP?

You can download PHP from any of the members of the PHP network of sites. These can be found at http://www.php.net/. You can also use anonymous CVS to get the absolute latest version of the source. For more information, go to http://cvs.php.net/.

2. Are pre-compiled binary versions available?

We only distribute precompiled binaries for Windows systems, as we are not able to compile PHP for every major Linux/Unix platform with every extension combination. Also note, that many Linux distributions come with PHP built in these days. Windows binaries can be downloaded from our Downloads (http://www.php.net/downloads.php) page, for Linux binaries, please visit your distributions website.

3. Where can I get libraries needed to compile some of the optional PHP extensions?

> **Anmerkung:** Those marked with * are not thread-safe libraries, and should not be used with PHP as a server module in the multi-threaded Windows web servers (IIS, Netscape). This does not matter in Unix environments, yet.

- LDAP (unix) (ftp://ftp.openldap.org/pub/openldap/openldap-stable.tgz).
- LDAP* (unix) (ftp://terminator.rs.itd.umich.edu/ldap/ldap-3.3.tar.Z).
- LDAP (unix/win) (http://developer.netscape.com/tech/directory/downloads.html) : Netscape Directory (LDAP) SDK 1.1.
- free LDAP server (http://developer.netscape.com/tech/directory/downloads.html).
- Berkeley DB2 (Unix/Win) (http://www.sleepycat.com/) : http://www.sleepycat.com/.
- SNMP* (Unix): (http://www.ece.ucdavis.edu/ucd-snmp/).
- GD* (Unix/Win) (http://www.boutell.com/gd/#buildgd).
- mSQL* (Win) (http://blnet.com/msqlpc/).
- mSQL* (Unix) (http://www.hughes.com.au/).
- PostgreSQL (Unix) (http://www.postgresql.org/).
- IMAP* (Win/Unix) (ftp://ftp.cac.washington.edu/imap/old/imap-4.5.tar.Z).
- Sybase-CT* (Linux, libc5) (http://www.php.net/extra/ctlib-linux-elf.tar.gz) : Available locally.
- FreeType (libttf): (http://www.freetype.org/).
- ZLib (Unix/Win32) (http://www.cdrom.com/pub/infozip/zlib/).
- expat XML parser (Unix/Win32) (http://www.jclark.com/xml/expat.html).
- PDFLib (http://www.pdflib.com/).
- mcrypt (ftp://argeas.cs-net.gr/pub/unix/mcrypt/).
- mhash (http://sasweb.de/mhash/).
- t1lib (http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/rmz/t1lib/t1lib.html).
- dmalloc (http://www.dmalloc.com/).
- aspell (http://download.sourceforge.net/aspell/aspell-.29.1.tar.gz).

- readline (ftp://prep.ai.mit.edu/pub/gnu/readline/).

4. How do I get these libraries to work?

You will need to follow instructions provided with the library. Some of these libraries are detected automatically when you run the 'configure' script of PHP (such as the GD library), and others you will have to enable using '`--with-EXTENSION`' options to '`configure`'. Run '`configure --help`' for a listing of these.

5. I got the latest version of the PHP source code from the CVS repository on my Windows machine, what do I need to compile it?

First, you will need Microsoft Visual C++ v6 (v5 may do it also, but we do it with v6), and you will need some support files. See the manual section about building PHP from source on Windows.

6. Where do I find the Browser Capabilities File?

You can find a `browscap.ini` file at http://www.cyscape.com/asp/browscap/.

Kapitel 46. Database issues