PHP Handbuch

Stig Sæther Bakken
Alexander Aulbach
Egon Schmid
Jim Winstead
Lars Torben Wilson
Rasmus Lerdorf
Zeev Suraski

Herausgegeben von

Egon Schmid

Inhaltsverzeichnis

Vor

Vorwort

Vorwort

Inhaltsverzeichnis

PHP ist die Abkürzung für "PHP: Hypertext Preprocessor" und ist 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 und DSSSL (Document Style and Semantics Specification Language) für die Formatierung. Die Tools zur Transformation in HTML, TeX und RTF sind Jade von James Clark und die Modularen DocBook Stylesheets von Norman Walsh. Die Programme zur Herstellung des PHP Handbuchs wurden von Stig Sæther Bakken ausgewählt.

Dieses HTML Handbuch wird jeden Tag neu erzeugt und kann bei http://snaps.php.net/manual/ bezogen werden.

I Einführung

Inhaltsverzeichnis

Kapitel 1 Einleitung

Inhaltsverzeichnis

Was ist PHP?

PHP (offiziell: "PHP: Hypertext Preprocessor") ist eine server-seitige, in HTML eingebettete Skriptsprache.

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

Beispiel 1-1 Ein einleitendes Beispiel

```
 1
 2 <html>
 3     <head>
 4         <title>Beispiel</title>
 5     </head>
 6     <body>
 7         <?php echo "Hallo, ich bin ein PHP-Skript!"; ?>
 8     </body>
 9 </html>
10
```

Dieser Skript unterscheidet sich von einem CGI-Skript, der in einer Sprache 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 zu 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 client-seitigen Sprache wie Javaskript unterscheidet ist, dass der Code vom Server ausgeführt wird. Sollten sie einen Skript wie den obigen auf ihrem Server ausführen, würde der Besucher nur das Ergebnis dessen empfangen, ohne die Möglichkeit zu haben, 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.

Kapitel 2 Installation

Inhaltsverzeichnis

Download der aktuellsten Version

Den Quellcode sowie die Binärdistributionen für die verschiedenen Plattformen (inklusive Windows) erhalten Sie unter `http://www.php.net/`.

Kapitel 3 Configuration

Inhaltsverzeichnis

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 dieses nur wenn der Server gestartet wird. Ist PHP als CGI-Version konfiguriert worden, dann geschieht dieses 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" Konfigurations-Einstellungen 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 automatisch in die aufgerufene Datei eingebettet, als wenn Sie die Datei mittels der **include()** Funktion eingebunden hätten. So wird include_path benutzt.

Der Wert none deaktiviert auto-appending.

> **Anmerkung:** Wird das Skript mit der Funktion **exit()**beendet, wird auto-append *not* aktiv.

auto_prepend_file string

Hier können Sie eine Datei angeben, die automatisch vor der Hauptdatei aufgerufen wird. Die Datei wird automatisch in die aufgerufene Datei eingebettet, als wenn Sie die Datei mittels der **include()** Funktion eingebunden hätten. So wird include_path benutzt.

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 auf einer per-directory oder per-virtual Server Basis aus bzw. einschalten wollen. Wenn Sie **`php3_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
2	normale Warnungen
4	Fehler des parsers, meistens Syntaxfehler
8	unkritische Warnungen, die ignoriert werden können. Sinnvoll in der Entwicklungsphase.

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

open_basedir string

Limitiert die Anzahl der Dateien, die von PHP in einem bestimmten Verzeichnis geöffnet werden können.

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 es nicht öffnen. Alle symbolischen Links sind hier mit eingeschlossen, so das es auch nicht möglich ist dieses Verbot mittels symlink zu umgehen.

Der Wert . gibt an, das 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 Standarteinstellung ist, alle Dateien zum parsen freizugeben.

gpc_order string

Beeinflußt die Reihenfolgen von GET/POST/COOKIE Variablen parsing. Die Standarteinstellung 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

Standartmäßig 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()** and **fopen_with_path()** Funktionen nach Dateien suchen. Das Format ist ähnlich den System's PATH Umgebungsvariablen. Eine Liste von Verzeichnissen, getrennt durch einen Doppelpunkt bei UNIX oder ein Semikolon bei WINDOWS.

Beispiel 3-1 UNIX include_path

```
1
2 include_path=.:/home/httpd/php-lib
3
```

Beispiel 3-2 Windows include_path

```
1
2 include_path=".;c:\www\phplib"
3
```

Der Standartwert 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 stehen, 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 backslashes 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, 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 Backslashes 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/ falsche Skripte vorbeugen. Bei komplexen Datenbankabfragen sollte man jedoch bedenken, das der Standartwert 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

`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 dieses Option aktiviert ist, wird der letzte Fehler immer in der Umgebungsvariablen $php_errormsg abgelegt sein.

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 schreibar 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-Concanetation benutzt wurde.

Mail Konfiguration Direktiven

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, wenn mails mit der **mail()** Funktion 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, das dieses nicht funktioniert, können Sie den Pfad hier eintragen.

Auf Systemen, wo Sendmail nicht benutzt wird, sollten sie -wenn vorhanden- diesen Pfad auf den Sendmail Wrapper/ Ersatz setzen. Zum Beispiel: Qmail Benutzer könne den Pfad normalerweise wie folgt setzen: `/var/qmail/bin/sendmail`.

Safe Mode Konfiguration Direktiven

safe_mode boolean

Wenn Sie diesen Schalter auf ON setzen, können Sie PHP im sicheren Modus betreiben. Mehr dazu erfahren Sie unter: Security chapter

safe_mode_exec_dir string

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

Debugger Konfiguration Direktiven

debugger.host string

DNS name oder IP Adresse des Host, der vom Debugger benutzt wird.

debugger.port string

Port Nummer, 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 PHP- Extensionen dynamisch mit der Funktion **dl()** aktivieren bzw. deaktivieren per Virtual Server or per Directory.

Der Haupgrund das dynamische Laden von extensionen zu deaktivieren ist Sicherheit. Wenn die Option aktiviert ist, ist es möglich, safe_mode und open_basedir Restriktionen zu ignorieren.

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

extension_dir string

Hier geben Sie das Verzeichnis an, indem PHP dynamisch zu ladenden Extensionen findet.

extension string

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

MySQL Konfigurations Direktiven

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 Standartbenutzers eingegeben (wird benutzt, falls kein anderer Name angegen wird).

mysql.default_password string

Hier wird das Standartpasswort 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 gesamten Verbindungen pro Prozess an (persistenten und normale).

mSQL Konfigurations Directives

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 gesamten Verbindungen pro Prozess an (persistenten und normale).

Postgres Konfigurations Direktiven

pgsql.allow_persistent boolean

Erlaubt persistente Postgres - Verbindungen.

pgsql.max_persistent integer

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

pgsql.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale).

Sybase Konfigurations Direktiven

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 gesamten Verbindungen pro Prozess an (persistenten und normale).

Sybase-CT Konfigurations Direktiven

sybct.allow_persistent boolean

Erlaubtr persistente Sybase-CT Verbindungen. Standartwert ist ON.

sybct.max_persistent integer

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

sybct.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale). Standartwert ist -1 (unbegrenzt).

sybct.min_server_severity integer

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

sybct.min_client_severity integer

Client library Nachrichten mit höherem oder gleichem Gewicht wie 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 Standartwert 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, das wenn die max_execution_time für einen Verbindungsversuch vorbei ist, Ihr Skript beendet wird, bevor es eine Fehlermeldung (Oder andere Aktion) ausgeben kann. Der Standartwert ist hier 1 Minute.

sybct.timeout integer

Hier können Sie die maximale Zeit in Sekunden angeben, die für einen erfolgreichen select_db oder andern DB-Query gewartet wird bis eine Fehlermeldung erscheint. Beachten Sie, das wenn die max_execution_time für einen DB-Query vorbei ist, Ihr Skript beendet wird, bevor es eine Fehlermeldung (Oder andere Aktion) ausgeben kann. Der Standartwert 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. Standartwert ist none.

Informix Konfigurations Direktiven

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 gesamten Verbindungen pro Prozess an (persistenten und normale).

ifx.default_host string

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

ifx.default_password string

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

ifx.blobinfile boolean

Setzten 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 vorangehende 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", 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 Konfigurations Direktiven

bcmath.scale integer

Anzahl der Nachkommastellen der bcmath-Funktion.

Browser Capability Konfigurations Direktiven

browscap string

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

Unified ODBC Configuration Directives

uodbc.default_db string

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

uodbc.default_user string

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

uodbc.default_pw string

ODBC Standart- Passwort, 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 gesamten ODBC- Verbindungen pro Prozess an (persistente und normale).

Kapitel 4 Sicherheit

Inhaltsverzeichnis

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 vielerlei Zwecke einsetzen kann, allerdings bedeutet es auch, dass es Kombinationen gibt, die zur Folge haben, daß die Installation nicht genug Sicherheit bietet. Dieses Kapitel beschreibt die verschiedenen Kombinationen der Konfigurationseinstellungen und unter welchen Gegebenheiten sie sicher genutzt werden können.

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 Scripts zu schaffen. Bei dieser Konfiguration wird das ausführbare PHP-Binary üblicherweise im cgi-bin Verzeichnis des Webservers installiert. CERT advisory CA-96.11 spricht sich gegen die Plazierung von Interpretern im cgi-bin Verzeichnis aus. Obwohl das PHP-Binary als standalone Interpreter verwendet werden kann, wurde PHP entwickelt, um Angriffe, die durch diese Konfiguration möglich werden, abzuwehren:

- Sytemdatenzugriff: `http://my.host/cgi-bin/php?/etc/passwd`

  Die Abfrageinformation in einer URL, die auf ein Fragezeichen (?) folgt, wird durch das CGI-Interface als Kommandozeilenargument an den Interpreter weitergereicht. Üblicherweise wird von Interpretern die Datei geöffnet und ausgeführt, die als erstes Argument in der Kommandozeile steht.

  Beim Aufruf als CGI-Binary verweigert PHP die Interpretierung der

Kommandozeilenargumente.

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

  Die Pfadinformation, Teil der URL nach dem PHP-Binarynamen, `/secret/doc.html` wurde ursprünglich 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.php3`. Unglücklicherweise wird, wenn der Aufruf bereits in dieser Form geschieht, vom Webserver keine Zugriffsüberprüfung der Datei `/secret/script.php3`, 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 kann beim compilieren die Konfigurationsoption --enable-force-cgi-redirect und zur Laufzeit die Direktive doc_root and 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: only public files served

Wenn der Server keine Inhalte hat, die durch Passwort oder IP-basierte Zugriffskontrolle geschützt sind, gibt es für diese Konfiguration keinen Grund. Wenn der Webserver keine Redirects erlaubt oder keine Möglichkeit hat, auf einer sicher umgeleiteten Anfrage mit dem PHP-Binary Verbindung aufzunehmen, kann die Option --disable-force-cgi-redirect im configure-Script angegeben werden. Nichtsdestotrotz müssen Sie sicherstellenn, daß Ihre PHP-Scripte nicht auf die eine oder anderen Art des Aufrufs angewiesen sind, weder direkt durch `http://my.host/cgi-bin/php/dir/script.php3` noch durch einen Redirect `http://my.host/dir/script.php3`.

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

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

Diese Option, die beim Kompilieren verwendet wird, verhindert grundsätzlich den Aufruf von PHP mit einer URL wie beispielsweise `http://my.host/cgi-bin/php/secretdir/script.php3`. 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:

```
1
2 Action php3-script /cgi-bin/php
3 AddHandler php3-script .php3
4
```

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 eine 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, sodern als reguläres HTML-Dokument angezeigt werden kann dies den Verlust von geistigem Eigentum und Sicherheit (Passwörter!) zur Folge haben. Von daher ziehen es viele Sysadmins vor, eine zweite Verzeichnisstruktur für Skripte, auf die nur durch das PHP-CGI zugegriffen werden soll, einzurichten. Diese werden dann stets interpretiert und nicht angezeigt.

Auch wenn die Methode, sicherzustellen dass die Anfragen nicht umgeleitet werden (wie im vorangegangenen Kapitel beschrieben), nicht verfügbar ist, ist es notwendig, ein doc_root für Scripts zusätzlich zum Dokumentenverzeichnis einzurichten.

Sie können Das PHP-Skriptverzeichnis durch den Eintrag doc_root in der Konfigurationsdatei ändern, oder Sie setzen die Umgebungsvariable `PHP_DOCUMENT_ROOT`. Wenn sie gesetzt ist, wird die CGI-Version von PHP den Namen der zu öffenenden Datei stets mit *`doc_root`* und der Pfadinformation der Anfrage zusammensetzen, so daß 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.php3` hat nicht zum Ergebnis, daß eine Datei im Home-Verzeichnis des Benutzers geöffnet wird, sondern eine Datei namens `~user/doc.php3` 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.php3` eine Aatei namens `doc.php3` 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.php3`.

Die *`user_dir`* Erweiterung erfolgte im Hinblick auf die *`doc_root`* Einstellung. So können Zugriffe auf das Dokumenten- und Benutzerverzeichnis 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 plazieren, beispielsweise in `/usr/local/bin`. Der einzige Nachteil dieses Verfahrens ist, dass eine Zeile, ähnlich der folgenden:

```
1
2 #!/usr/local/bin/php
3
```

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 `PATH_INFO` und `PATH_TRANSLATED` Informationen korrekt auswertet, sollte der PHP-Parser mit der Option --enable-discard-path kompiliert werden.

II Sprachreferenz

Inhaltsverzeichnis

Kapitel 5 Grundlagen der Syntax

Inhaltsverzeichnis

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
  2 1.  <? echo ("Das ist die einfachste Moeglichkeit:eine SGML-´proccessing
instruction´\n"); ?>
  3
  4 2.  <?php echo("Um XML-konforme Dokumente herzustellen, benutzt man diese
Syntax\n"); ?>
  5
  6 3.  <script language="php">
  7         echo ("Manche Editoren (z. B. Frontpage) moegen keine ´processing
instructions´\n");
  8     </script>
  9
 10 4.  <% echo ("Optional können auch Tags im ASP-Stil verwendet werden"); %>
 11     <%= $variable; # Das ist ein Abkuerzung fuer "<%echo .." %>
 12
```

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 folgende Zeilenschaltung mit ein, falls diese vorhanden ist.

Kapitel 6 Typen

Inhaltsverzeichnis

PHP unterstützt die folgenden Typ-Deklarationen:

- Array
- Fließkomma-Zahl
- Integer
- Object
- String / Zeichenkette

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.

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

qBeachten Sie, dass eine Variable je nach Gebrauch und Situation auf unterschiedliche Art und Weise typisiert sein kann. Weitere Informationen sehen sie unter Typ-Veränderung.

Integer-Typen

Integer-Typen können durch Gebrauch einer der folgenden Zuweisungsarten angegeben werden:

```
1
2 $a = 1234; # Zahlenwert im dezimal-Format
3 $a = -123; # ein negativer Zahlenwert
4 $a = 0123; # Oktal-Zahl (83 im dezimal-Format) Achtung:
5            # Fehlerquelle bei Strings mit führenden Nullen!
6 $a = 0x12; # Zahlenwert im hexadezimal-Format
7            # (entspricht 18 im dezimal-Format)
8
```

Kapitel 7 Variablen

Inhaltsverzeichnis

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 die 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_\xzf-\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).

```
 1
 2 $var = "Du";
 3 $vaR = "und";
 4 $Var = "ich";
 5 $vAr = "wir lernen PHP"
 6 echo "$var $vaR $Var, $vAr"; // gibt "Du und ich, wir lernen PHP" aus
 7
 8 $4site  = 'nicht jetzt';     // ungültig, da Anfang eine Zahl
 9 $_4site = 'nicht jetzt';     // gültig, da Unterstrich am Anfang
10 $täbyte = 'irgendwas';       // gültig, da 'ä' dem ASCII-Wert 228 entspricht
11
```

Variablen werden in PHP3 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.

PHP4 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 PHP3. 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 "&" plus "amp;" (&) der (Ausgangs-, Quell-) Variablen voranstellen, die sie einer anderen Variablen zuweisen wollen. Der folgende Skript- Ausschnitt wird zweimal 'Mein Name ist Bob' ausgeben:

```
 1
 2 <?php
 3 $foo = 'Bob';              // 'Bob' der Variablen $foo zuweisen.
 4 $bar = &$foo;          // Zeiger auf $foo in $bar erzeugen.
 5 $bar = "My name is $bar"; // $bar verändern...
 6 echo $foo;                // $foo wurde dadurch ebenfalls verändert.
 7 echo $bar;
 8 ?>
 9
```

Zu beachten ist, dass nur Variablenbezeichner referenziert werden können.

```
 1
 2 <?php
 3 $foo = 25;
 4 $bar = &$foo;     // Gültige Zuweisung.
 5 $bar = &(24 * 7); // Ungültig, da kein Variablenbezeichner
 6                        zugewiesen wird.
 7 function test() {
 8     return 25;
 9 }
10
11 $bar = &test();   // Ungültig.
12 ?>
13
```

Zurück | Anfang | Vor
Typen-Tricks | Hoch | Vordefinierte Variablen

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

```
1
2 <?php
3 define("CONSTANT", "Hallo Welt.");
4 echo CONSTANT; // gibt  "Hallo Welt." aus.
5 ?>
6
```

Beispiel 8-2 Die Benutzung von __FILE__ und __LINE__

```
1
2 <?php
3 function report_error($file, $line, $message) {
4     echo "Ein Fehler ist aufgetreten in $file in Zeile $line: $message.";
5 }
6
7 report_error(__FILE__,__LINE__, "Irgendetwas ist falsch gelaufen!");
8 ?>
9
```

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:

```
1
2 function foo () {
3     return 5;
4 }
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:

```
 1
 2 $eins ? $zwei : $drei
 3
```

Wenn der Wert des ersten Sub-Ausdruckes (hier: $eins) wahr ist (d. h. nicht NULL), dann wird der Wert des zweiten Subausdrucks (hier: $zwei) zurückgegeben und ist das Ergebnis des konditionalen Ausdrucks. Andernfalls (d. h. wenn der erste Ausdruck falsch ist), wird der Wert des dritten Subausdruckes (hier: $drei) zurückgegeben.

Das folgende Beispiel sollte das Verständnis von Prä- und Post-Inkrement und von Ausdrücken im allgemeinen erleichtern:

```
 1
 2 function verdoppeln($i) {
 3     return $i*2;
 4 }
 5 $b = $a = 5;        /* Weise den Variablen $a und $b beiden den Wert 5 zu */
 6 $c = $a++;          /* Post-Inkrement, der urspruengliche Wert von $a (also 5)
 7                        wird $c zugewiesen. */
 8 $e = $d = ++$b;     /* Prae-Inkrement, der erhöhte Wert von $b (= 6) wird $d und
 9                        $e zugewiesen. */
10
11 /* An diesem Punkt sind $d und $e beide gleich 6 */
12
13 $f = verdoppeln($d++);  /* Weise $f den doppelten Wert von $d vor
14                        der Erhöhung um eins zu, 2*6 = 12 */
15 $g = double(++$e);  /* Weise $g den doppelten Wert von $e nach
16                        der Erhoehung zu, 2*7 = 14 to $g */
17 $h = $g += 10;      /* Zuerst wie $g um 10 erhöht und hat schliesslich den Wert
18                        24. Der Wert dieser Zuweisung (24) wird dann $h
zugewiesen,
19                        womit $h ebenfalls den Wert von 24 hat. */
20
```

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.

Zurück	Anfang	Vor
Konstanten	Hoch	Operators

Kapitel 10 Operators

Inhaltsverzeichnis

Arithmetic Operators

Remember basic arithmetic from school? These work just like those.

Tabelle 10-1 Arithmetic Operators

example	name	result
$a + $b	Addition	Sum of $a and $b.
$a - $b	Subtraction	Difference of $a and $b.
$a * $b	Multiplication	Product of $a and $b.
$a / $b	Division	Quotient of $a and $b.
$a % $b	Modulus	Remainder of $a divided by $b.

Kapitel 11 Kontroll-Strukturen

Inhaltsverzeichnis

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 von C ist:

```
1
2 if (expr)
3     Anweisung
4
```

Wie im Abschnitt über Expressions / Ausdrücke beschrieben, wird expr auf seinen wirklichen Wertinhalt ausgewertet. Wenn *`expr`* 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*:

```
1
2 if ($a > $b)
3     print "a ist grösser als b";
4
```

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:

```
1
2 if ($a > $b) {
3     print "a ist grösser als b";
4     $b = $a;
5 }
6
```

If-Anweisungen können ohne Einschränkung innerhalb anderer `if`-Anweisungen definiert werden. Das ermöglicht ihnen völlige Flexibilität bei der bedingten Ausführung verschiedenster Programmteile.

Zurück: String Operators | Anfang | Hoch | Vor: `else`

Kapitel 12 Funktionen

Inhaltsverzeichnis

Vom Nutzer definierte Funktionen

Eine Funktion kann wie folgt definiert werden:

```
1
2 function foo ($arg_1, $arg_2, ..., $arg_n) {
3     echo "Beispielfunktion.\n";
4     return $retval;
5 }
6
```

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.

Kapitel 13 Klassen und Objekte

Inhaltsverzeichnis

Klassen

Eine Klasse ist eine Sammlung von Variablen und von Funktionen, die mit diesen Variablen arbeiten. Eine Klasse wird folgendermaßen definiert:

```
 1
 2 <?php
 3 class Einkaufswagen {
 4     var $produkte;  // Produkte in Ihrem Einkaufswagen
 5
 6     // Füge dem Einkaufswagen $anzahl Artikel der Sorte $artnr zu
 7
 8     function fuege_hinzu ($artnr, $anzahl) {
 9         $this->produkte[$artnr] += $anzahl;
10     }
11
12     // Nimm $anzahl von Artikel wieder aus dem Einkaufswagen
13
14     function nimm_heraus ($artnr, $anzahl) {
15         if ($this->produkte[$artnr] > $anzahl) {
16             $this->produkte[$artnr] -= $anzahl;
17             return true;
18         } else {
19             return false;
20         }
21     }
22 }
23 ?>
24
```

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.

```
 1
 2 $einkaufswagen = new Einkaufswagen;
 3 $einkaufswagen->fuege_hinzu("10", 1);
 4
```

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.

```
 1
 2 class Mein_Einkaufswagen extends Einkaufswagen {
 3     var $besitzer;
 4
 5     function setze_besitzer ($name) {
 6         $this->besitzer = $name;
 7     }
 8 }
 9
```

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.

```
 1
 2 $meinkaufswagen = new Mein_Einkaufswagen; // Kreiere einen Einkaufswagen
 3 $meinkaufswagen->setze_besitzer ("kris"); // Name dieser Klasse
 4 print $meinkaufswagen->besitzer;          // schreibe den Namen des Besitzers
 5 $meinkaufswagen->fuege_hinzu ("10", 1);   // (Siehe oben, vererbt
 6                                           // von Einkaufswagen)
 7
```

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 Kontruktor, wenn Sie den gleichen Namen wie die Klasse trägt.

```
 1
 2 class Auto_Einkaufswagen extends Einkaufswagen {
 3     function Auto_Einkaufswagen () {
 4         $this->fuege_hinzu ("10", 1);
 5     }
 6 }
 7
```

Die Klasse Auto_Einkaufswagen entspricht der Klasse Einkaufswagen plus einen Kontstruktor, der bereits für eine erste Füllung (1 Artikel der Nummer 10) gesorgt hat. Jeder neu erzeugte Auto_Einkaufswagen enthält so von vorneherein diesen Artikel. Kontruktoren können auch Parameter enthalten. Aber diese Parameter sind optional, und können so nützlicher eingesetzt werden.

```
 1
 2 class Konstruktor_Einkaufswagen extends Einkaufswagen {
 3     function Konstruktor_Einkaufswagen ($produkt = "10", $anzahl = 1) {
 4         $this->fuege_hinzu ($produkt, $anzahl);
 5     }
 6 }
 7
 8 // Kaufe wieder den gleichen alten Kram ein.
 9
10 $standard_einkaufswagen   = new Konstruktor_Einkaufswagen;
11
12 // Kaufe etwas bestimmtes ein ...
13
14 $anderer_Einkaufswagen = new Konstruktor_Einkaufswagen ("20", 17);
15
```

Achtung
Bei abgeleiteten Klassen wird der Konstruktor der Ursprungsklasse nicht automatisch aufgerufen, wenn der Konstruktor der abgeleiteten Klasse aufgerufen wird.

III Features

Inhaltsverzeichnis

Kapitel 14 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 aufgeführeten 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 error_reporting Option 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 15 Creating GIF images

PHP ist nicht darauf beschränkt, nur HTML auszugeben. Es kann auch dazu genutzt werden, einzelne GIF-Bilddateien oder auch auf bequeme Art und Weise eine ganze Abfolge von GIF´s zu erzeugen. Um all das zu tun, müssen sie nur die GD-Library mit Bildfunktionen eincompilieren.

Beispiel 15-1 GIF-Erzeugung mit PHP

```
 1
 2 <?php
 3     Header("Content-type: image/gif");
 4     $string=implode($argv," ");
 5     $im = imagecreatefromgif("images/taste1.gif");
 6     $orange = ImageColorAllocate($im, 220, 210, 60);
 7     $px = (imagesx($im)-7.5*strlen($string))/2;
 8     ImageString($im,3,$px,9,$string,$orange);
 9     ImageGif($im);
10     ImageDestroy($im);
11 ?>
12
```

Dieses Beispiel würde von einer Seite mit einem Tag wie diesem aufgerufen: <img src="taste.php3?text">. Der obige taste.php3-Skript nimmt dann den "text"-String und legt ihn über das Grundbild, was in diesem Fall "images/taste1.gif" ist, 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 werden sie dynamisch generiert.

Kapitel 16 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 **Header()** benutzen, um eine "Authentifizierung notwendig"-Nachricht 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 die grundlegende Authentifizierung unterstützt. Näheres hierzu bei der **Header()** Funktion.

Ein Auszug aus einem Skript, das die Clientauthentifizierung auf einer Seite erzwingt, würde so aussehen:

Beispiel 16-1 HTTP Authentifizierung

```
 1
 2 <?php
 3   if(!isset($PHP_AUTH_USER)) {
 4     Header("WWW-Authenticate: Basic realm=\"My Realm\"");
 5     Header("HTTP/1.0 401 Unauthorized");
 6     echo "Text to send if user hits Cancel button\n";
 7     exit;
 8   } else {
 9     echo "Hello $PHP_AUTH_USER.<P>";
10     echo "You entered $PHP_AUTH_PW as your password.<P>";
11   }
12 ?>
13
```

Anstatt $PHP_AUTH_USE und $PHP_AUTH_PW einfach nur auszugeben, wird man den Bunutzernamen 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 zu schaffen.

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 Zugrifskontrolle zu identifizieren.

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 Cache des Authentifizierungsfensters, wenn der Server eine 401-Meldung zurückgibt. Dies kann benutzt werden, um einen Benutzer "auszuloggen" und eine erneute Eingabe des Benutzernamens/Passworts zu erzwingen. Einige nutzen dies, um Logins nach Ablauf einer bestimmten Zeitspanne auslaufen zu lassen oder einen Logout-Button zur Verfügung zu stellen.

Beispiel 16-2 HTTP Authentifizierung, mit erneuter Anforderung von Name/Passwort

```
 1
 2 <?php
 3   function  authenticate()  {
 4     Header( "WWW-authenticate:  basic  realm='Test  Authentication  System'");
 5     Header( "HTTP/1.0  401  Unauthorized");
 6     echo  "You  must  enter  a  valid  login  ID  and  password  to  access  this
resource\n";
 7     exit;
 8   }
 9
10   if(!isset($PHP_AUTH_USER)  ||  ($SeenBefore ==  1  &&  !strcmp($OldAuth,
$PHP_AUTH_USER))  )  {
11     authenticate();
12   }
13   else  {
14     echo  "Welcome:  $PHP_AUTH_USER<BR>";
15     echo  "Old:  $OldAuth";
16     echo  "<FORM  ACTION=\"$PHP_SELF\"  METHOD=POST>\n";
17     echo  "<INPUT  TYPE=HIDDEN  NAME=\"SeenBefore\"  VALUE=\"1\">\n";
18     echo  "<INPUT  TYPE=HIDDEN  NAME=\"OldAuth\"  VALUE=\"$PHP_AUTH_USER\">\n";
19     echo  "<INPUT  TYPE=Submit  VALUE=\"Re  Authenticate\">\n";
20     echo  "</FORM>\n";
21
22 }
23 ?>
24
```

Dieses Verhalten wird vom HTTP Basic A-uthentication Standard nicht gefordert, daher sollte man sich nie darauf verlassen. Tests mit Lynx haben gezeigt, daß Lynx die Authentifizierungsinformationen nicht bei Erhalt einer 401-Meldung 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 Server mit der CGI-Version von PHP aufgrund einer Einschränkung des IIS nicht funktioniert.

Zurück	Anfang	Vor
Creating GIF images	Hoch	Cookies

Kapitel 17 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 eines 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 **setcookie()** Funktion.

Kapitel 18 Steuerung von Dateiuploads

Inhaltsverzeichnis

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 18-1 Formular für den Dateiupload

```
1
2 <FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
3 <INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
4 Send this file: <INPUT NAME="userfile" TYPE="file">
5 <INPUT TYPE="submit" VALUE="Send File">
6 </FORM>
7
```

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 erfogreichem 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 temorären Verzeichnis gelöscht, sobald das Skript abgearbeitet ist, wenn sie nicht verschoben oder umbenannt wurde.

Kapitel 19 Using remote files

As long as support for the "URL fopen wrapper" is enabled when you configure PHP (which it is unless you explicitly pass the `--disable-url-fopen-wrapper` flag to configure), you can use HTTP and FTP URLs with most functions that take a filename as a parameter, including the **require()** and **include()** statements.

Anmerkung: You can't use remote files in **include()** and **require()** statements on Windows.

For example, you can use this to open a file on a remote web server, parse the output for the data you want, and then use that data in a database query, or simply to output it in a style matching the rest of your website.

Beispiel 19-1 Getting the title of a remote page

```
 1
 2 <?php
 3   $file = fopen("http://www.php.net/", "r");
 4   if (!$file) {
 5     echo "<p>Unable to open remote file.\n";
 6     exit;
 7   }
 8   while (!feof($file)) {
 9     $line = fgets($file, 1024);
10     /* This only works if the title and its tags are on one line. */
11     if (eregi("<title>(.*)</title>", $line, $out)) {
12       $title = $out[1];
13       break;
14     }
15   }
16   fclose($file);
17 ?>
18
```

You can also write to files on an FTP as long you connect as a user with the correct access rights, and the file doesn't exist already. To connect as a user other than 'anonymous', you need to specify the username (and possibly password) within the URL, such as 'ftp://user:password@ftp.example.com/path/to/file'. (You can use the same sort of syntax to access files via HTTP when they require Basic authentication.)

Beispiel 19-2 Storing data on a remote server

```
 1
 2 <?php
 3   $file = fopen("ftp://ftp.php.net/incoming/outputfile", "w");
 4   if (!$file) {
 5     echo "<p>Unable to open remote file for writing.\n";
 6     exit;
 7   }
 8   /* Write the data here. */
 9   fputs($file, "$HTTP_USER_AGENT\n");
10   fclose($file);
11 ?>
12
```

Anmerkung: You might get the idea from the example above to use this technique to write to a remote log, but as mentioned above, you can only write to a new file using the URL fopen() wrappers. To do distributed logging like that, you should take a look at **syslog()**.

Zurück | Anfang | Vor
PUT-Unterstützung | Hoch | Verbindungssteuerung

Kapitel 20 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.

Man kann entscheiden, ob der Verbindungsabbruch seitens eines 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 registierte 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.

Zurück | Anfang | Vor
Using remote files | Hoch | Persistente Datenbankverbindungen

Kapitel 21 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 leistungsstartke 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 con 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') koodiniert, 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.

IV Funktionsreferenz

Inhaltsverzeichnis

I. Apache-spezifische Funktionen

Inhaltsverzeichnis

II. Mathematische Funktionen mit beliebiger Genauigkeit

Diese Funktionen sind nur verfügbar, wenn PHP mit `--enable-bcmath` konfiguriert wurde.

Inhaltsverzeichnis

III. Array Funktionen

Inhaltsverzeichnis

pos Get the current element from an array

prev Rewind the internal array pointer

range Create an array containing a range of integers

reset Set the internal pointer of an array to its first element

rsort Sort an array in reverse order

shuffle Shuffle an array

sizeof Get the number of elements in an array

sort Sort an array

uasort Sort an array with a user-defined comparison function and maintain index association

uksort Sort an array by keys using a user-defined comparison function

usort Sort an array by values using a user-defined comparison function

---

IV. Aspell Funktionen

Die **aspell()** Funktionen erlauben es ein Wort auf korrekte Rechtschreibung zu prüfen und Alternativen anzubieten.

Sie benötigen die aspell-Bibliothek von: http:/metalab.unc.edu/kevina/aspell/.

Inhaltsverzeichnis

V. Kalender-Funktionen

Dieses Modul bietet eine Reihe von Funktionen, die die Umwandlung von Daten zwischen verschiedenen Kalendern erleichtern. Die gemeinsame Basis für diese Umwandlung bilden 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.

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.

Inhaltsverzeichnis

VI. COM support functions for Windows

These functions are only available on the Windows version of PHP. These functions have been added in PHP4.

Inhaltsverzeichnis

VII. Klassen- und Objekt Funktionen

Inhaltsverzeichnis

VIII. ClibPDF functions

ClibPDF lets you create PDF documents with PHP. It is available at FastIO but it isn't free software. You should definitely read the licence before you start playing with ClibPDF. If you cannot fullfil the licence agreement consider using pdflib by Thomas Merz, which is also very powerful. ClibPDF functionality and API is similar to Thomas Merz's pdflib but, according to FastIO, ClibPDF is faster and creates smaller documents. This may have changed with the new version 2.0 of pdflib. A simple benchmark (the pdfclock.c example from pdflib 2.0 turned into a php script) actually shows no difference in speed at all. The file size is also similar if compression is turned off. So, try them both and see which one does the job for you.

This documentation should be read alongside the ClibPDF manual since it explains the library in much greater detail.

Many functions in the native ClibPDF and the PHP module, as well as in pdflib, have the same name. All functions except for **cpdf_open()** take the handle for the document as their first parameter. Currently this handle is not used internally since ClibPDF does not support the creation of several PDF documents at the same time. Actually, you should not even try it, the results are unpredictable. I can't oversee what the consequences in a multi threaded environment are. According to the author of ClibPDF this will change in one of the next releases (current version when this was written is 1.10). If you need this functionality use the pdflib module.

> **Anmerkung:** The function **cpdf_set_font()** has changed since PHP3 to support asian fonts. The encoding parameter is no longer an integer but a string.

One big advantage of ClibPDF over pdflib is the possibility to create the pdf document completely in memory without using temporary files. It also provides the ability to pass coordinates in a predefined unit length. This is a handy feature but can be simulated with **pdf_translate()**.

Most of the functions are fairly easy to use. The most difficult part is probably creating a very simple PDF document at all. The following example should help you get started. It creates a document with one page. The page contains the text "Times-Roman" in an outlined 30pt font. The text is underlined.

Beispiel 1 Simple ClibPDF Example

```
 1
 2 <?php
 3 $cpdf = cpdf_open(0);
 4 cpdf_page_init($cpdf, 1, 0, 595, 842);
 5 cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
 6 cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
 7 cpdf_set_text_rendering($cpdf, 1);
 8 cpdf_text($cpdf, "Times Roman outlined", 50, 750);
 9 cpdf_moveto($cpdf, 50, 740);
10 cpdf_lineto($cpdf, 330, 740);
11 cpdf_stroke($cpdf);
12 cpdf_finalize($cpdf);
13 Header("Content-type: application/pdf");
14 cpdf_output_buffer($cpdf);
15 cpdf_close($cpdf);
16 ?>
17
```

The pdflib distribution contains a more complex example which creates a series of pages with an analog clock. Here is that example converted into PHP using the ClibPDF extension:

Beispiel 2 pdfclock example from pdflib 2.0 distribution

```
 1
 2 <?php
 3 $radius = 200;
 4 $margin = 20;
 5 $pagecount = 40;
 6
 7 $pdf = cpdf_open(0);
 8 cpdf_set_creator($pdf, "pdf_clock.php3");
 9 cpdf_set_title($pdf, "Analog Clock");
10
11 while($pagecount-- > 0) {
12   cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius +
$margin), 1.0);
13
14   cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */
15
16   cpdf_translate($pdf, $radius + $margin, $radius + $margin);
17   cpdf_save($pdf);
18   cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
19
20   /* minute strokes */
21   cpdf_setlinewidth($pdf, 2.0);
22   for ($alpha = 0; $alpha < 360; $alpha += 6)
23     {
24     cpdf_rotate($pdf, 6.0);
25     cpdf_moveto($pdf, $radius, 0.0);
26     cpdf_lineto($pdf, $radius-$margin/3, 0.0);
27     cpdf_stroke($pdf);
28     }
29
30   cpdf_restore($pdf);
31   cpdf_save($pdf);
32
33   /* 5 minute strokes */
34   cpdf_setlinewidth($pdf, 3.0);
35   for ($alpha = 0; $alpha < 360; $alpha += 30)
36   {
37     cpdf_rotate($pdf, 30.0);
38     cpdf_moveto($pdf, $radius, 0.0);
39     cpdf_lineto($pdf, $radius-$margin, 0.0);
40     cpdf_stroke($pdf);
41   }
42
43   $ltime = getdate();
44
45   /* draw hour hand */
46   cpdf_save($pdf);
47   cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
48   cpdf_moveto($pdf, -$radius/10, -$radius/20);
49   cpdf_lineto($pdf, $radius/2, 0.0);
50   cpdf_lineto($pdf, -$radius/10, $radius/20);
51   cpdf_closepath($pdf);
52   cpdf_fill($pdf);
53   cpdf_restore($pdf);
54
55   /* draw minute hand */
56   cpdf_save($pdf);
57   cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) *
6.0);
```

```
58   cpdf_moveto($pdf, -$radius/10, -$radius/20);
59   cpdf_lineto($pdf, $radius * 0.8, 0.0);
60   cpdf_lineto($pdf, -$radius/10, $radius/20);
61   cpdf_closepath($pdf);
62   cpdf_fill($pdf);
63   cpdf_restore($pdf);
64
65   /* draw second hand */
66   cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
67   cpdf_setlinewidth($pdf, 2);
68   cpdf_save($pdf);
69   cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
70   cpdf_moveto($pdf, -$radius/5, 0.0);
71   cpdf_lineto($pdf, $radius, 0.0);
72   cpdf_stroke($pdf);
73   cpdf_restore($pdf);
74
75   /* draw little circle at center */
76   cpdf_circle($pdf, 0, 0, $radius/30);
77   cpdf_fill($pdf);
78
79   cpdf_restore($pdf);
80
81   cpdf_finalize_page($pdf, $pagecount+1);
82 }
83
84 cpdf_finalize($pdf);
85 Header("Content-type: application/pdf");
86 cpdf_output_buffer($pdf);
87 cpdf_close($pdf);
88 ?>
89
```

Inhaltsverzeichnis

cpdf_set_text_rendering Determines how text is rendered
cpdf_set_horiz_scaling Sets horizontal scaling of text
cpdf_set_text_rise Sets the text rise
cpdf_set_text_matrix Sets the text matrix
cpdf_set_text_pos Sets text position
cpdf_set_char_spacing Sets character spacing
cpdf_set_word_spacing Sets spacing between words
cpdf_continue_text Output text in next line
cpdf_stringwidth Returns width of text in current font
cpdf_save Saves current enviroment
cpdf_restore Restores formerly saved enviroment
cpdf_translate Sets origin of coordinate system
cpdf_scale Sets scaling
cpdf_rotate Sets rotation
cpdf_setflat Sets flatness
cpdf_setlinejoin Sets linejoin parameter
cpdf_setlinecap Sets linecap aparameter
cpdf_setmiterlimit Sets miter limit
cpdf_setlinewidth Sets line width
cpdf_setdash Sets dash pattern
cpdf_moveto Sets current point
cpdf_rmoveto Sets current point
cpdf_curveto Draws a curve
cpdf_lineto Draws a line
cpdf_rlineto Draws a line
cpdf_circle Draw a circle
cpdf_arc Draws an arc
cpdf_rect Draw a rectangle
cpdf_closepath Close path
cpdf_stroke Draw line along path
cpdf_closepath_stroke Close path and draw line along path
cpdf_fill Fill current path
cpdf_fill_stroke Fill and stroke current path
cpdf_closepath_fill_stroke Close, fill and stroke current path
cpdf_clip Clips to current path
cpdf_setgray_fill Sets filling color to gray value
cpdf_setgray_stroke Sets drawing color to gray value
cpdf_setgray Sets drawing and filling color to gray value
cpdf_setrgbcolor_fill Sets filling color to rgb color value
cpdf_setrgbcolor_stroke Sets drawing color to rgb color value
cpdf_setrgbcolor Sets drawing and filling color to rgb color value
cpdf_add_outline Adds bookmark for current page
cpdf_set_page_animation Sets duration between pages
cpdf_import_jpeg Opens a JPEG image

IX. Cybercash payment functions

These functions are only available if the interpreter has been compiled with the `--with-cybercash=[DIR]`. These functions have been added in PHP4.

Inhaltsverzeichnis

X. DOM XML functions

These functions are only available if PHP was configured with `--with-dom=[DIR]`, using the GNOME xml library. These functions have been added in PHP4.

This module defines the following constants:

Tabelle 1 XML constants

Constant	Value	Description
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	

This module defines a number of classes. The DOM XML functions return a parsed tree of the XML document with each node being an object belonging to one of these classes.

Inhaltsverzeichnis

XI. Compression functions

This module uses the functions of 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).

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

```
 1
 2 <?php
 3   $filename = tempnam('/tmp', 'zlibtest').'.gz';
 4   print "<html>\n<head></head>\n<body>\n<pre>\n";
 5   $s = "Only a test, test, test, test, test, test, test, test!\n";
 6   // open file for writing with maximum compression
 7   $zp = gzopen($filename, "w9");
 8   // write string to file
 9   gzwrite($zp, $s);
10   // close file
11   gzclose($zp);
12   // open file for reading
13   $zp = gzopen($filename, "r");
14   // read 3 char
15   print gzread($zp, 3);
16   // output until end of the file and close it.
17   gzpassthru($zp);
18   print "\n";
19   // open file and print content (the 2nd time).
20   if (readgzfile($filename) != strlen($s)) {
21           echo "Error with zlib functions!";
22   }
23   unlink($filename);
24   print "</pre>\n</h1></body>\n</html>\n";
25 ?>
26
```

Inhaltsverzeichnis

gzfile read entire gz-file into an array

gzgetc get character from gz-file pointer

gzgets get line from file pointer

gzgetss get line from gz-file pointer and strip HTML tags

gzopen open gz-file

gzpassthru output all remaining data on a gz-file pointer

gzputs write to a gz-file pointer

gzread Binary-safe gz-file read

gzrewind rewind the position of a gz-file pointer

gzseek seek on a gz-file pointer

gztell tell gz-file pointer read/write position

gzwrite Binary-safe gz-file write

readgzfile output a gz-file

XII. Database (dbm-style) abstraction layer functions

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a subset of features modern databases such as Sleepycat Software's DB2 support. (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

The behaviour of various aspects depend on the implementation of the underlying database. Functions such as **dba_optimize()** and **dba_sync()** will do what they promise for one database and will do nothing for others.

The following handlers are supported:

- dbm is the oldest (original) type of Berkeley DB style databases. You should avoid it, if possible. We do not support the compatibility functions built into DB2 and gdbm, because they are only compatible on the source code level, but cannot handle the original dbm format.
- ndbm is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated).
- gdbm is the GNU database manager.
- db2 is Sleepycat Software's DB2. It is described as "a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications."
- cdb is "a fast, reliable, lightweight package for creating and reading constant databases." It is from the author of qmail and can be found here. Since it is constant, we support only reading operations.

Beispiel 1 DBA example

```
 1
 2 <?php
 3
 4 $id = dba_open("/tmp/test.db", "n", "db2");
 5
 6 if(!$id) {
 7     echo "dba_open failed\n";
 8     exit;
 9 }
10
11 dba_replace("key", "This is an example!", $id);
12
13 if(dba_exists("key", $id)) {
```

```
14     echo dba_fetch("key", $id);
15     dba_delete("key", $id);
16 }
17
18 dba_close($id);
19 ?>
20
```

DBA is binary safe and does not have any arbitrary limits. It inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to **dba_open()** or **dba_popen()**.

You can access all entries of a database in a linear way by using the **dba_firstkey()** and **dba_nextkey()** functions. You may not change the database while traversing it.

Beispiel 2 Traversing a database

```
 1
 2 <?php
 3
 4 # ...open database...
 5
 6 $key = dba_firstkey($id);
 7
 8 while($key != false) {
 9     if(...) { # remember the key to perform some action later
10         $handle_later[] = $key;
11     }
12     $key = dba_nextkey($id);
13 }
14
15 for($i = 0; $i < count($handle_later); $i++)
16     dba_delete($handle_later[$i], $id);
17
18 ?>
19
```

Inhaltsverzeichnis

dba_popen Open database persistently

dba_open Open database

dba_optimize Optimize database

dba_replace Replace or insert entry

dba_sync Synchronize database

XIII. Datums- und Zeit-Funktionen

Inhaltsverzeichnis

XIV. 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.

Inhaltsverzeichnis

XV. dbm functions

These functions allow you to store records stored in a dbm-style database. This type of database (supported by the Berkeley db, gdbm, and some system libraries, as well as a built-in flatfile library) stores key/value pairs (as opposed to the full-blown records supported by relational databases).

Beispiel 1 dbm example

```
 1
 2 $dbm = dbmopen("lastseen", "w");
 3 if (dbmexists($dbm, $userid)) {
 4   $last_seen = dbmfetch($dbm, $userid);
 5 } else {
 6   dbminsert($dbm, $userid, time());
 7 }
 8 do_stuff();
 9 dbmreplace($dbm, $userid, time());
10 dbmclose($dbm);
11
```

Inhaltsverzeichnis

XVI. Verzeichnis-Funktionen

Inhaltsverzeichnis

XVII. Dynamisch geladene Bibliothek

Inhaltsverzeichnis

XVIII. Encryption functions

These functions work using mcrypt.

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".

To use it, download libmcrypt-x.x.tar.gz from here and follow the included installation instructions. You need to compile PHP with the `--with-mcrypt` parameter to enable this extension.

mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. 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 in ECB mode

```
1
2 <?php
3 $key = "this is a very secret key";
4 $input = "Let us meet at 9 o'clock at the secret place.";
5
6 $encrypted_data = mcrypt_ecb(MCRYPT_TripleDES, $key, $input, MCRYPT_ENCRYPT);
7 ?>
8
```

This example will give you the encrypted data as a string in `$encrypted_data`.

Mcrypt can operate in four cipher modes (CBC, OFB, CFB, and ECB). 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) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated.

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 is that you can access the cipher from PHP with MCRYPT_ciphername.

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_BLOWFISH
- MCRYPT_DES
- MCRYPT_TripleDES
- MCRYPT_ThreeWAY
- MCRYPT_GOST
- MCRYPT_CRYPT

- MCRYPT_DES_COMPAT
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_CAST128
- MCRYPT_TEAN
- MCRYPT_RC2
- MCRYPT_TWOFISH (for older mcrypt 2.x versions)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_RC6
- MCRYPT_IDEA

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).

Inhaltsverzeichnis

XIX. filePro functions

These functions allow read-only access to data stored in filePro databases.

filePro is a registered trademark of Fiserv, Inc. You can find more information about filePro at http://www.fileproplus.com/.

Inhaltsverzeichnis

XX. Funktionen des Dateisystems

Inhaltsverzeichnis

fpassthru Gibt alle verbleibenden Daten eines Dateizeigers direkt aus.

fputs Schreibt Daten an die Position des Dateizeigers

fread Liest Binär-Dateien aus einer Datei

fseek Positioniert den Dateizeiger um einen Offset 'offset' vor oder zurück

ftell Ermittelt die aktuelle Position des Dateizeigers

fwrite Schreibt Binärdaten in eine Datei

set_file_buffer Setzt die Dateipufferung für einen gegebenen Dateizeiger

is_dir Ermittelt, ob der gegebene Dateiname ein Verzeichnis ist

is_executable Ermittelt, ob eine Datei ausführbar ist

is_file Ermittelt, ob der Dateiname eine reguläre Datei ist

is_link Ermittelt, ob der Dateiname ein synmbolischer Link ist

is_readable Ermittelt ob eine Datei lesbar ist

is_writeable Ermittelt, ob in eine Datei geschrieben werden kann

link Erzeugt einen absoluten Link

linkinfo Ermittelt Informationen über einen Link

mkdir Erstellt ein Verzeichnis

pclose Schließt einen Prozess-Dateizeiger

popen Öffnet einen Prozesszeiger

readfile Gibt eine Datei aus

readlink Gibt das Ziel eines symbolischen Links zurück

rename Benennt eine Datei um

rewind Setzt den Dateizeiger auf das erste Byte der Datei

rmdir Löscht ein Verzeichnis

stat Ermittelt diverse Informationen über eine Datei

lstat Ermittelt Informationen über einen symbolischen Link.

symlink Erzeugt einen symbolischen Link

tempnam Erzeugt einen eindeutigen Dateinamen

touch Setzt das Datum der letzten Änderung einer Datei

umask Ändert die aktuelle umask (Zugriffsrechte)

unlink Löscht eine Datei

XXI. Forms Data Format functions

Forms Data Format (FDF) is a format for handling forms within PDF documents. You should read the documentation at http://partners.adobe.com/asn/developer/acrosdk/forms.html for more information on what FDF is and how it is used in general.

> **Anmerkung:** Currently Adobe only provides a libc5 compatible version for Linux. Tests with glibc2 resulted in a segmentation fault. If somebody is able to make it work, please comment on this page.
>
> **Anmerkung:** If you run into problems configuring php with fdftk support, check whether the header file FdfTk.h and the library libFdfTk.so are at the right place. They should be in fdftk-dir/include and fdftk-dir/lib. This will not be the case if you just unpack the FdfTk distribution.

The general idea of FDF is similar to HTML forms. The diffence is basically the format how filled in data is transmitted to the server when the submit button is pressed (this is actually the Form Data Format) and the format of the form itself (which is the Portable Document Format, PDF). Processing the FDF data is one of the features provided by the fdf functions. But there is more. One may as well take an existing PDF form and populated the input fields with data without modifying the form itself. In such a case one would create a FDF document (**fdf_create()**) set the values of each input field (**fdf_set_value()**) and associate it with a PDF form (**fdf_set_file()**). Finally it has to be sent to the browser with MimeType application/vnd.fdf. The Acrobat reader plugin of your browser recognizes the MimeType, reads the associated PDF form and fills in the data from the FDF document.

The following examples shows just the evaluation of form data.

Beispiel 1 Evaluating a FDF document

```
 1
 2 <?php
 3 // Save the FDF data into a temp file
 4 $fdffp = fopen("test.fdf", "w");
 5 fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
 6 fclose($fdffp);
 7
 8 // Open temp file and evaluate data
 9 // The pdf form contained several input text fields with the names
10 // volume, date, comment, publisher, preparer, and two checkboxes
11 // show_publisher and show_preparer.
12 $fdf = fdf_open("test.fdf");
13 $volume = fdf_get_value($fdf, "volume");
14 echo "The volume field has the value '<B>$volume</B>'<BR>";
15
16 $date = fdf_get_value($fdf, "date");
17 echo "The date field has the value '<B>$date</B>'<BR>";
18
19 $comment = fdf_get_value($fdf, "comment");
```

```
20 echo "The comment field has the value '<B>$comment</B>'<BR>";
21
22 if(fdf_get_value($fdf, "show_publisher") == "On") {
23   $publisher = fdf_get_value($fdf, "publisher");
24   echo "The publisher field has the value '<B>$publisher</B>'<BR>";
25 } else
26   echo "Publisher shall not be shown.<BR>";
27
28 if(fdf_get_value($fdf, "show_preparer") == "On") {
29   $preparer = fdf_get_value($fdf, "preparer");
30   echo "The preparer field has the value '<B>$preparer</B>'<BR>";
31 } else
32   echo "Preparer shall not be shown.<BR>";
33 fdf_close($fdf);
34 ?>
35
```

Inhaltsverzeichnis

XXII. 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.

Inhaltsverzeichnis

XXIII. GNU Gettext

Inhaltsverzeichnis

XXIV. Hash functions

These functions are intended to work with mhash.

This is an interface to the mhash library. mhash supports a wide variety of hash algorithms such as MD5, SHA1, GOST, and many others.

To use it, download the mhash distribution from its web site and follow the included installation instructions. You need to compile PHP with the `--with-mhash` parameter to enable this extension.

mhash can be used to create checksums, message digests, and more.

Beispiel 1 Compute the SHA1 key and print it out as hex

```
1
2 <?php
3 $input = "Let us meet at 9 o' clock at the secret place.";
4 $hash = mhash(MHASH_SHA1, $input);
5
6 print "The hash is ".bin2hex($hash)."\n";
7
8 ?>
9
```

This will produce:

```
1
2 The hash is d3b85d710d8f6e4e5efd4d5e67d041f9cecedafe
3
```

For a complete list of supported hashes, refer to the documentation of mhash. The general rule is that you can access the hash algorithm from PHP with MHASH_HASHNAME. For example, to access HAVAL you use the PHP constant MHASH_HAVAL.

Here is a list of hashes which are currently supported by mhash. If a hash is not listed here, but is listed by mhash as supported, you can safely assume that this documentation is outdated.

- MHASH_MD5
- MHASH_SHA1
- MHASH_HAVAL
- MHASH_RIPEMD160
- MHASH_RIPEMD128
- MHASH_SNEFRU
- MHASH_TIGER

- MHASH_GOST
- MHASH_CRC32
- MHASH_CRC32B

Inhaltsverzeichnis

XXV. HTTP-Funktionen

Diese Funktionen ermöglichen die Manipulation der an den Browser geschickten Informationen bis hinunter auf HTTP-Protokoll-Ebene.

Inhaltsverzeichnis

XXVI. Hyperwave functions

Introduction

Hyperwave has been developed at IICM 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. 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 (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:

```
1
2 RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
3
```

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:

```
1
2 RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
3
```

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

```
1
2 RewriteEngine on
3
```

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.

Inhaltsverzeichnis

hw_GetAnchorsObj object records of anchors of document

hw_Mv moves objects

hw_Identify identifies as user

hw_InCollections check if object ids in collections

hw_Info info about connection

hw_InsColl insert collection

hw_InsDoc insert document

hw_InsertDocument upload any document

hw_InsertObject inserts an object record

hw_mapid Maps global id on virtual local id

hw_Modifyobject modifies object record

hw_New_Document create new document

hw_Objrec2Array convert attributes from object record to object array

hw_OutputDocument prints hw_document

hw_pConnect make a persistent database connection

hw_PipeDocument retrieve any document

hw_Root root object id

hw_Unlock unlock object

hw_Who List of currently logged in users

hw_Username name of currently logged in user

XXVII. Grafik-Funktionen

Sie können die Grafik-Funktionen von PHP nicht nur benutzen, um die Grösse von JPEG, GIF und PNG-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.

Inhaltsverzeichnis

ImageFillToBorder Flächen-Farbfüllung ("flood fill") mit einer angegebenen Farbe

ImageFontHeight Ermittelt die Font-Höhe

ImageFontWidth Ermittelt die Font-Breite

ImageGIF Ausgabe eines Bildes an den Browser oder in eine Datei

ImageJPEG Ausgabe des Bildes im Browser oder als Datei

ImageInterlace Schaltet die Interlaced-Darstellung eines Bildes an oder aus

ImageLine Zeichnen einer Linie

ImageLoadFont Lädt einen neuen Font

ImagePolygon Zeichne ein Vieleck (Polygon)

ImagePSBBox Ermittelt die Ausmaße des Rechtecks, das für die Ausgabe eines Textes unter Verwendung eines PostScript-Fonts (Typ 1) notwendig ist

ImagePSEncodeFont Ändert die Vektor-Beschreibung eines Fonts

ImagePSFreeFont Gibt den durch einen Typ 1 PostScript-Font belegten Speicher wieder frei

ImagePSLoadFont Lädt einen Typ 1 PostScript-Font aus einer Datei

ImagePSText Ausgabe eines Textes auf einem Bild unter Verwendung von Typ 1 PostScript-Fonts

ImageRectangle Zeichnet ein Rechteck

ImageSetPixel Setzt ein einzelnes Pixel

ImageString Zeichnet einen horizontalen String

ImageStringUp Zeichnet einen vertikalen Strin

ImageSX Ermittelt die Bild-Breite

ImageSY Ermittelt die Bild-Höhe

ImageTTFBBox Ermittelt die Rahmenmaße für die Ausgabe eines Textes im True-Type-Format

ImageTTFText Erzeugt TTF-Text im Bild

XXVIII. IMAP, POP3 und NNTP Funktionen

Dieses Packet 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 -Protokol 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 : Simple Mail Transfer Protocol (SMTP)
- RFC822 : Standard for ARPA internet text messages
- RFC2060 : Internet Message Access Protocol (IMAP) Version 4rev1
- RFC1939 : Post Office Protocol Version 3 (POP3)
- RFC977 : Network News Transfer Protocol (NNTP)
- RFC2076 : Common Internet Message Headers
- RFC2045 , RFC2046 , RFC2047 , RFC2048 & RFC2049 : Multipurpose Internet Mail Extensions (MIME)

Eine ausführliche Übersicht bietet auch das Buch Programming Internet Email von David Wood.

Inhaltsverzeichnis

imap_delete Merkt eine Nachricht des aktuellen Postfachs zum Löschen vor.

imap_deletemailbox Löscht ein Postfach

imap_expunge Löscht alle zum Löschen vorgemerkten Nachrichten

imap_fetchbody Liefert einen bestimmten Abschnitt aus dem Körper einer Nachricht.

imap_fetchstructure Liefert die Struktur der angegebenen Nachricht

imap_header Liefert den Kopf einer Nachricht

imap_headers Liefert eine Zusammenfassung aller Nachrichtenköpfe eines Postfachs

imap_listmailbox Liefert eine Liste der Postfach-Namen

imap_getmailboxes Liefert detailierte Informationen über eine Auswahl von Postfächern

imap_listsubscribed Liefert eine Liste aller abbonierten Postfächer

imap_getsubscribed Liefert eine Auswahl aller abbonierten Postfächer

imap_mail_copy Kopiert Nachrichten in ein Postfach

imap_mail_move Verschiebt Nachrichten in ein anderes Postfach

imap_num_msg Liefert die Anzahl der Nachrichten im aktuellen Postfach

imap_num_recent Liefert die Anzahl der neu hinzugekommenen Nachrichten im aktuellen Postfach

imap_open Öffnet eine Verbindung zu einem Postfach auf einem POP-, IMAP- oder NNTP-Server

imap_ping Prüft, ob die aktuelle Verbindung noch verfügbar ist

imap_renamemailbox Ändert den Namen eines Postfachs

imap_reopen Wechselt das aktuelle Postfach der Verbindung

imap_subscribe Abbonieren eines Postfachs

imap_undelete Nimmt eine bereits gesetzte Löschmarkierung einer Nachricht zurück

imap_unsubscribe Abbonement eines Postfachs beenden

imap_qprint Konvertiert einen quoted-printable kodierten String in einen 8bit-String

imap_8bit Konvertiert einen String in einen quoted-printable kodierten String.

imap_binary Konvertiert 8Bit-Text in einen BASE64-kodierten String.

imap_scanmailbox Druchsucht Postfächer nach einen String

imap_mailboxmsginfo Liefert Informationen über das aktuelle Postfach

imap_rfc822_write_address Bildet aus Realnamen, Postfach und Server eine korekt formatierte Mail-Adresse

imap_rfc822_parse_adrlist Parsen eines Adress-Strings

imap_setflag_full Setzt Flags einer Nachricht

imap_clearflag_full Löscht Flags einer Nachricht

imap_sort Sortiert Nachrichten eines Postfachs

imap_fetchheader Liefert den Kopf einer Nachricht

imap_uid Liefert die UID zu einem gegeben Nachrichten-Index

imap_msgno Liefert den aktuellen Nachrichten-Index zu einer UID

imap_search Sucht Nachrichten, die den übergebenen Suchkriterien entsprechen

imap_last_error Diese Funktion liefert die zuletzt während dieses Seitenzugriffs aufgetretene Fehlermeldung (falls vorhanden)

imap_errors Diese Funktion liefert alle bisher aufgetretenen Fehlermeldungen

imap_alerts Diese Funktion liefert alle alert-Meldungen zurück, die während der Ausführung dieses Requests bzw. seit dem letzten Reset des alert-Stacks aufgetreten sind.

imap_status Liefert ausgewählte Statusinformationen zum angegebenen Postfach

imap_utf7_decode Dekodiert einem String im modifizierten UTF-7 Format.

imap_utf7_encode Kodiert Text im modifizieren UTF-7 Format

imap_utf8 Konvertiert Text zu UTF8

imap_fetch_overview Liefert einen Auszug aus den Header-Feldern von Nachrichten

imap_mail_compose Erzeugt eine MIME-Nachricht aus gegebenen Header- und Body-Teilen

imap_mail Versendet eine Email

XXIX. 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 overide 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.

Inhaltsverzeichnis

ifx_close Close Informix connection

ifx_query Send Informix query

ifx_prepare Prepare an SQL-statement for execution

ifx_do Execute a previously prepared SQL-statement

ifx_error Returns error code of last Informix call

ifx_errormsg Returns error message of last Informix call

ifx_affected_rows Get number of rows affected by a query

ifx_getsqlca Get the contents of sqlca.sqlerrd[0..5] after a query

ifx_fetch_row Get row as enumerated array

ifx_htmltbl_result Formats all rows of a query into a HTML table

ifx_fieldtypes List of Informix SQL fields

ifx_fieldproperties List of SQL fieldproperties

ifx_num_fields Returns the number of columns in the query

ifx_num_rows Count the rows already fetched a query

ifx_free_result Releases resources for the query

ifx_create_char Creates an char object

ifx_free_char Deletes the char object

ifx_update_char Updates the content of the char object

ifx_get_char Return the content of the char object

ifx_create_blob Creates an blob object

ifx_copy_blob Duplicates the given blob object

ifx_free_blob Deletes the blob object

ifx_get_blob Return the content of a blob object

ifx_update_blob Updates the content of the blob object

ifx_blobinfile_mode Set the default blob mode for all select queries

ifx_textasvarchar Set the default text mode

ifx_byteasvarchar Set the default byte mode

ifx_nullformat Sets the default return value on a fetch row

ifxus_create_slob Creates an slob object and opens it

ifx_free_slob Deletes the slob object

ifxus_close_slob Deletes the slob object

ifxus_open_slob Opens an slob object

ifxus_tell_slob Returns the current file or seek position

ifxus_seek_slob Sets the current file or seek position

ifxus_read_slob Reads nbytes of the slob object

ifxus_write_slob Writes a string into the slob object

XXX. InterBase functions

InterBase is a popular database put out by Borland/Inprise. More information about InterBase is available at http://www.interbase.com/. Oh, by the way, Interbase just joined the open source movement!

Inhaltsverzeichnis

XXXI. LDAP functions

Introduction to LDAP

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure.

The concept is similar to your hard disk directory structure, except that in this context, the root directory is "The world" and the first level subdirectories are "countries". Lower levels of the directory structure contain entries for companies, organisations or places, while yet lower still we find directory entries for people, and perhaps equipment or documents.

To refer to a file in a subdirectory on your hard disk, you might use something like

```
/usr/local/myapp/docs
```

The forwards slash marks each division in the reference, and the sequence is read from left to right.

The equivalent to the fully qualified file reference in LDAP is the "distinguished name", referred to simply as "dn". An example dn might be.

```
cn=John Smith,ou=Accounts,o=My Company,c=US
```

The comma marks each division in the reference, and the sequence is read from right to left. You would read this dn as ..

```
country = US
organization = My Company
organizationalUnit = Accounts
commonName = John Smith
```

In the same way as there are no hard rules about how you organise the directory structure of a hard disk, a directory server manager can set up any structure that is meaningful for the purpose. However, there are some conventions that are used. The message is that you can not write code to access a directory server unless you know something about its structure, any more than you can use a database without some knowledge of what is available.

Complete code example

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Beispiel 1 LDAP search example

```
 1
 2 <?php
 3 // basic sequence with LDAP is connect, bind, search, interpret search
 4 // result, close connection
 5
 6 echo "<h3>LDAP query test</h3>";
 7 echo "Connecting ...";
 8 $ds=ldap_connect("localhost");  // must be a valid LDAP server!
 9 echo "connect result is ".$ds."<p>";
10
11 if ($ds) {
12     echo "Binding ...";
13     $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
14                            // read-only access echo "Bind result is
15     echo "Bind result is ".$r."<p>";
16
17     echo "Searching for (sn=S*) ...";
18     // Search surname entry
19     $sr=ldap_search($ds,"o=My Company, c=US", "sn=S*");
20     echo "Search result is ".$sr."<p>";
21
22     echo "Number of entires returned is ".ldap_count_entries($ds,$sr)."<p>";
23
24     echo "Getting entries ...<p>";
25     $info = ldap_get_entries($ds, $sr);
26     echo "Data for ".$info["count"]." items returned:<p>";
27
28     for ($i=0; $i<$info["count"]; $i++) {
29         echo "dn is: ". $info[$i]["dn"] ."<br>";
30         echo "first cn entry is: ". $info[$i]["cn"][0] ."<br>";
31         echo "first email entry is: ". $info[$i]["mail"][0] ."<p>";
32     }
33
34     echo "Closing connection";
35     ldap_close($ds);
36
37 } else {
38     echo "<h4>Unable to connect to LDAP server</h4>";
39 }
40 ?>
41
```

Using the PHP LDAP calls

You will need to get and compile LDAP client libraries from either the University of Michigan ldap-3.3 package or the Netscape Directory SDK. You will also need to recompile PHP with LDAP support enabled before PHP's LDAP calls will work.

Before you can use the LDAP calls you will need to know ..

- The name or address of the directory server you will use
- The "base dn" of the server (the part of the world directory that is held on this server, which could be "o=My Company,c=US")
- Whether you need a password to access the server (many servers will provide read access for an "anonymous bind" but require a password for anything else)

The typical sequence of LDAP calls you will make in an application will follow this pattern:

```
ldap_connect()    // establish connection to server
   |
ldap_bind()       // anonymous or authenticated "login"
   |
do something like search or update the directory
and display the results
   |
ldap_close()      // "logout"
```

More Information

Lots of information about LDAP can be found at

- Netscape
- University of Michigan
- OpenLDAP Project
- LDAP World

The Netscape SDK contains a helpful Programmer's Guide in .html format.

Inhaltsverzeichnis

ldap_read Read an entry

ldap_search Search LDAP tree

ldap_unbind Unbind from LDAP directory

ldap_err2str Convert LDAP error number into string error message

ldap_errno Return the LDAP error number of the last LDAP command

ldap_error Return the LDAP error message of the last LDAP command

---

XXXII. Mail Funktionen

Die **Mail()** Funktionen erlauben das Versenden von Mail.

Inhaltsverzeichnis

XXXIII. Mathematische Funktionen

Introduction

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.

Inhaltsverzeichnis

base_convert Konvertiert Werte zwischen Zahlensystemen unterschiedlicher Basis

BinDec Binär zu dezimal Konvertierung

Ceil Aufrunden

Cos Cosinus

DecBin Dezimal zu binär Konvertierung

DecHex Dezimal zu hexadezimal Konvertierung

DecOct Dezimal zu oktal Konvertierung

deg2rad Grad in Bogenmaß wandeln

Exp e hoch ...

Floor Abrunden

getrandmax Größtmöglicher Zufallswert

HexDec Hexadezimal zu dezimal Konvertierung

Log Natürlicher Logarithmus

Log10 Zehner-Logarithmus

max Maximalwert bestimmen

min Minimalwert bestimmen

mt_rand Erzeugt 'bessere' Zufallszahlen

mt_srand seed the better random number generator

mt_getrandmax show largest possible random value

number_format Formartiert eine Zahl mit Tausendergruppen

OctDec Oktal zu dezimal Konvertierung

pi Wert von PI

pow Exponentialfunktion

rad2deg Grad in Bogenmaß wandeln

rand (Pseudo-) Zufallszahl generieren

round Runden eines Wertes

Sin Sinus

Sqrt Quadratwurzel

srand Startwert für den Zufallsgenerator festlegen

Tan Tangens

XXXIV. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It's written to be very modular, with plugable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and reoccuring events.

With libmcal, central calendar servers can be accessed and used, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `--with-mcal`. That requires the mcal library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

The following constants are defined when using the MCAL module: 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. Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

Inhaltsverzeichnis

mcal_event_init Initializes a streams global event structure
mcal_event_set_category Sets the category of the streams global event structure
mcal_event_set_title Sets the title of the streams global event structure
mcal_event_set_description Sets the description of the streams global event structure
mcal_event_set_start Sets the start date and time of the streams global event structure
mcal_event_set_end Sets the end date and time of the streams global event structure
mcal_event_set_alarm Sets the alarm of the streams global event structure
mcal_event_set_class Sets the class of the streams global event structure
mcal_is_leap_year Returns if the given year is a leap year or not
mcal_days_in_month Returns the number of days in the given month
mcal_date_valid Returns true if the given year, month, day is a valid date
mcal_time_valid Returns true if the given year, month, day is a valid time
mcal_day_of_week Returns the day of the week of the given date
mcal_day_of_year Returns the day of the year of the given date
mcal_date_compare Compares two dates
mcal_next_recurrence Returns the next recurrence of the event
mcal_event_set_recur_none Sets the recurrence of the streams global event structure
mcal_event_set_recur_daily Sets the recurrence of the streams global event structure
mcal_event_set_recur_weekly Sets the recurrence of the streams global event structure
mcal_event_set_recur_monthly_mday Sets the recurrence of the streams global event structure
mcal_event_set_recur_monthly_wday Sets the recurrence of the streams global event structure
mcal_event_set_recur_yearly Sets the recurrence of the streams global event structure
mcal_fetch_current_stream_event Returns an object containing the current streams event structure

---

XXXV. Microsoft SQL Server functions

Inhaltsverzeichnis

XXXVI. Sonstige Funktionen

Diese Funktionen stehen hier, weil sie in keine andere Kategorie passen.

Inhaltsverzeichnis

XXXVII. mSQL functions

Inhaltsverzeichnis

XXXVIII. MySQL Funktionen

Diese Funktionen erlauben den Zugriff auf einen MySQL Datenbank-Server.

Weiter Informationen zu MySQL gibt es unter http://www.mysql.com/.

In diesem Dokument werden ein 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 impliziet als auch expliziet 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 weiters 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.

Inhaltsverzeichnis

mysql_fetch_field Liefert ein Objekt mit Feldinformationen aus einem Anfrageergebnis

mysql_fetch_lengths Liefert die Länge eines jeden Feldes in einem Datensatz

mysql_fetch_object Liefert einen Datensatz als Objekt

mysql_fetch_row Liefert einen Datensatz als indiziertes Array

mysql_field_name Liefert den Namen eines Feldes in einem Anfrageergebnis

mysql_field_seek Setzt den Feldzeiger auf ein bestimmtes Feld

mysql_field_table Liefert den Namen der Tabelle, die das genannte Feld enthält

mysql_field_type Liefert den Typ eines Feldes in einem Anfrageergebnis

mysql_field_flags Liefert die Flags eines Feldes in einem Anfrageergebnis

mysql_field_len Liefert die Länge eines Feldes

mysql_free_result Entfernt ein Ergebnis

mysql_insert_id Liefert die Kennung einer vorherigen INSERT-Operation

mysql_list_fields Listet die Felder einer Tabelle auf

mysql_list_dbs Liefert eine Liste der verfügbaren Datenbanken auf dem Server

mysql_list_tables Liefert eine Liste der Tabellen in einer Datenbank

mysql_num_fields Liefert die Anzahl der Felder in einem Ergebnis

mysql_num_rows Liefert die Anzahl der Datensätze im Ergebnis

mysql_pconnect Öffnete eine persistente Verbindung zum MySQL Server

mysql_query Sendet eine SQL Anfrage zum Datenbankserver

mysql_result Liefert Ergebnis

mysql_select_db Wählt eine Datenbank aus

mysql_tablename Liefert den Namen einer Tabelle

XXXIX. Netzwerk Funktionen

Inhaltsverzeichnis

XL. NIS functions

NIS (formerly called Yellow Pages) allows network management of important administrative files (e.g. the password file). For more information refer to the NIS manpage and Introduction to YP/NIS. There is also a book called Managing NFS and NIS by Hal Stern.

To get these functions to work, you have to configure PHP with `--with-yp`.

Inhaltsverzeichnis

XLI. ODBC Funktionen

Inhaltsverzeichnis

XLII. Oracle functions

Inhaltsverzeichnis

XLIII. Oracle 8 functions

These functions allow you to access Oracle8 and Oracle7 databases. It uses the Oracle8 Call-Interface (OCI8). You will need the Oracle8 client libraries to use this extension.

This extension is more flexible than the standard Oracle extension. It supports binding of global and local PHP variables to Oracle placeholders, has full LOB, FILE and ROWID support and allows you to use user-supplied define variables.

Inhaltsverzeichnis

OCIColumnName Returns the name of a column.

OCIColumnType Returns the data type of a column.

OCIParse Parse a query and return a statement

OCIError Return the last error of stmt|conn|global. If no error happened returns false.

OCIInternalDebug Enables or disables internal debug output. By default it is disabled

---

Zurück Anfang Vor
Ora_Rollback Hoch OCIDefineByName

XLIV. PDF Funktionen

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 und die TIFF Bibliothek, 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.

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.

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. Es hat eine geringfügig andere Programmierschnittstelle. Schauen Sie in die Funktionsübersicht für weitere Details.

Zur Zeit werden alle Versionen von pdflib unterstützt. Es wird empfohlen, dass die neuste Version Verwendung findet, weil nur dann alle Möglichkeiten genutzt werden können und Probleme mit älteren Versionen umgangen werden. Leider hat der Umstieg von 0.6 auf 2.x so weitreichende Änderungen hervorgerufen, dass einige der PHP Funktionen geändert werden mussten. Hier ist ein Liste der davon betroffen Funktionen:

- Die Info-Struktur existiert nicht mehr. Deshalb ist auch die Funktion **pdf_get_info()** nicht mehr notwendig und die Funktionen **pdf_set_info_creator()**, **pdf_set_info_title()**, **pdf_set_info_author()**, **pdf_set_info_subject()** und **pdf_set_info_keywords()** benötigen nicht mehr die Info-Struktur als ersten Parameter sondern das PDF-Dokument. Dies bedeutet zudem, dass das PDF-Dokument vor dem Aufruf dieser Funktion geöffnet werden muss.
- Die Art und Weise wie ein neues PDF-Dokument geöffnet wird hat sich verändert. Die Funktion **pdf_open()** benötigt nur noch einen Parameter, den Deskriptor der mit **fopen()** geöffneten Datei.

Mit der Version 2.01 von pdflib kamen weitere Veränderungen hinzu, die weitgehend von PHP abgefangen wurden. Dennoch werden einige Funktionen nicht mehr benötigt (z. B. **pdf_put_image()**). Diese Funktionen liefern bei Aufruf eine harmlose Warnung.

Das PDF Module von PHP verwendet zwei neue Typen von Variablen (bei Verwendung von pdflib 2.x ist es nur einer). Sie werden *pdfdoc* und *pdfinfo* (*pdfinfo* existiert in pdflib 2.x nicht) genannt. *pdfdoc* ist ein Zeiger auf das PDF Dokument und wird bei fast allen Funktion als erster Parameter erwartet. *pdfinfo* enthält Meta-Daten über das PDF Dokument. Sie muss vor dem Aufruf von **pdf_open()** gesetzt werden.

> **Anmerkung:** Der folgende Absatz ist nur gültig bei Verwendung von pdflib 0.6. Lesen sie die Dokumentation von pdflib bei Verwendung neuerer Versionen.

Zur Textausgabe in ein PDF-Dokument müssen die Zeichensatz-Metriken (afm Dateien) für jeden Zeichensatz bereitgestellt werden. Afm-Dateien enthalten die Geometrie eines jeden Buchstaben in einem Postscript-Zeichensatz. In der Grundeinstellung werden diese Dateien in dem Verzeichnis 'fonts', relativ zum Verzeichnis in dem das PHP-Skript ist, gesucht. Wie gesagt, dies galt für pdflib 0.6, neuere Versionen brauchen für gewöhnlich diese Dateien nicht.

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. Es benutzt die PHP-Funktionen basierend auf pdflib 0.6. Dieses Skript erstellt die PDF-Datei test.pdf, welche nur aus einer Seite besteht. Auf der Seite befindet sich der Text "Times-Roman" in einem outlined 30pt Zeichensatz. Der Text ist zudem unterstrichen.

Beispiel 1 Erstellung eines PDF Dokuments mit pdflib 0.6

```
 1
 2 <?php
 3 $fp = fopen("test.pdf", "w");
 4 $info = PDF_get_info();
 5 pdf_set_info_author($info, "Uwe Steinmann");
 6 PDF_set_info_title($info, "Test for PHP wrapper of PDFlib 0.6");
 7 PDF_set_info_author($info, "Name of Author");
 8 pdf_set_info_creator($info, "See Author");
 9 pdf_set_info_subject($info, "Testing");
10 $pdf = PDF_open($fp, $info);
11 PDF_begin_page($pdf, 595, 842);
12 PDF_add_outline($pdf, "Page 1");
13 pdf_set_font($pdf, "Times-Roman", 30, 4);
14 pdf_set_text_rendering($pdf, 1);
15 PDF_show_xy($pdf, "Times Roman outlined", 50, 750);
16 pdf_moveto($pdf, 50, 740);
17 pdf_lineto($pdf, 330, 740);
18 pdf_stroke($pdf);
19 PDF_end_page($pdf);
20 PDF_close($pdf);
21 fclose($fp);
22 echo "<A HREF=getpdf.php3>finished</A>";
23 ?>
24
```

Das PHP-Skript getpdf.php3 liefert nur das PDF-Document.

```
 1
 2 <?php
 3 $fp = fopen("test.pdf", "r");
 4 header("Content-type: application/pdf");
 5 fpassthru($fp);
 6 fclose($fp);
 7 ?>
 8
```

Das gleiche Beispiel mit pdflib 2.x sieht wie folgt aus:

Beispiel 2 Erstellung eines PDF-Document mit pdflib 2.x

```
 1
 2 <?php
 3 $fp = fopen("test.pdf", "w");
 4 $pdf = PDF_open($fp);
 5 pdf_set_info_author($pdf, "Uwe Steinmann");
 6 PDF_set_info_title($pdf, "Test for PHP wrapper of PDFlib 2.0");
 7 PDF_set_info_author($pdf, "Name of Author");
 8 pdf_set_info_creator($pdf, "See Author");
 9 pdf_set_info_subject($pdf, "Testing");
10 PDF_begin_page($pdf, 595, 842);
11 PDF_add_outline($pdf, "Page 1");
12 pdf_set_font($pdf, "Times-Roman", 30, 4);
13 pdf_set_text_rendering($pdf, 1);
14 PDF_show_xy($pdf, "Times Roman outlined", 50, 750);
15 pdf_moveto($pdf, 50, 740);
16 pdf_lineto($pdf, 330, 740);
17 pdf_stroke($pdf);
18 PDF_end_page($pdf);
19 PDF_close($pdf);
20 fclose($fp);
21 echo "<A HREF=getpdf.php3>finished</A>";
22 ?>
23
```

Das PHP-Skript getpdf.php3 ist zu dem oben beschriebenen identisch.

Die pdflib-Distribution enthält ein kompliezierteres 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 3 pdfclock Beispiel aus der pdflib 2.x Distribution

```
 1
 2 <?php
 3 $pdffilename = "clock.pdf";
 4 $radius = 200;
 5 $margin = 20;
 6 $pagecount = 40;
 7
 8 $fp = fopen($pdffilename, "w");
 9 $pdf = pdf_open($fp);
10 pdf_set_info_creator($pdf, "pdf_clock.php3");
11 pdf_set_info_author($pdf, "Uwe Steinmann");
12 pdf_set_info_title($pdf, "Analog Clock");
13
14 while($pagecount-- > 0) {
15     pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));
16
17     pdf_set_transition($pdf, 4);  /* wipe */
18     pdf_set_duration($pdf, 0.5);
19
20     pdf_translate($pdf, $radius + $margin, $radius + $margin);
21     pdf_save($pdf);
22     pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
23
24     /* minute strokes */
25     pdf_setlinewidth($pdf, 2.0);
26     for ($alpha = 0; $alpha < 360; $alpha += 6) {
```

```
27          pdf_rotate($pdf, 6.0);
28          pdf_moveto($pdf, $radius, 0.0);
29          pdf_lineto($pdf, $radius-$margin/3, 0.0);
30          pdf_stroke($pdf);
31      }
32
33      pdf_restore($pdf);
34      pdf_save($pdf);
35
36      /* 5 minute strokes */
37      pdf_setlinewidth($pdf, 3.0);
38      for ($alpha = 0; $alpha < 360; $alpha += 30) {
39          pdf_rotate($pdf, 30.0);
40          pdf_moveto($pdf, $radius, 0.0);
41          pdf_lineto($pdf, $radius-$margin, 0.0);
42          pdf_stroke($pdf);
43      }
44
45      $ltime = getdate();
46
47      /* draw hour hand */
48      pdf_save($pdf);
49      pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
50      pdf_moveto($pdf, -$radius/10, -$radius/20);
51      pdf_lineto($pdf, $radius/2, 0.0);
52      pdf_lineto($pdf, -$radius/10, $radius/20);
53      pdf_closepath($pdf);
54      pdf_fill($pdf);
55      pdf_restore($pdf);
56
57      /* draw minute hand */
58      pdf_save($pdf);
59      pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
60      pdf_moveto($pdf, -$radius/10, -$radius/20);
61      pdf_lineto($pdf, $radius * 0.8, 0.0);
62      pdf_lineto($pdf, -$radius/10, $radius/20);
63      pdf_closepath($pdf);
64      pdf_fill($pdf);
65      pdf_restore($pdf);
66
67      /* draw second hand */
68      pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
69      pdf_setlinewidth($pdf, 2);
70      pdf_save($pdf);
71      pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
72      pdf_moveto($pdf, -$radius/5, 0.0);
73      pdf_lineto($pdf, $radius, 0.0);
74      pdf_stroke($pdf);
75      pdf_restore($pdf);
76
77      /* draw little circle at center */
78      pdf_circle($pdf, 0, 0, $radius/30);
79      pdf_fill($pdf);
80
81      pdf_restore($pdf);
82
83      pdf_end_page($pdf);
84  }
```

```
85
86 $pdf = pdf_close($pdf);
87 fclose($fp);
88 echo "<A HREF=getpdf.php3?filename=".$pdffilename.">finished</A>";
89 ?>
90
```

Das PHP-Skript getpdf.php3 liefert nur das Dokument.

```
 1
 2 <?php
 3 $fp = fopen($filename, "r");
 4 header("Content-type: application/pdf");
 5 fpassthru($fp);
 6 fclose($fp);
 7 ?>
 8
```

Inhaltsverzeichnis

PDF_scale Setzt den Skalierungsfaktor
PDF_rotate Setzt die Rotation
PDF_setflat Sets flatness
PDF_setlinejoin Setzt die Verbindungsart von Linien
PDF_setlinecap Setzt den Type der Linienenden
PDF_setmiterlimit Sets miter limit
PDF_setlinewidth Setzt die Linienbreite
PDF_setdash Setzt das Muster für gestrichelte Linien
PDF_moveto Setzt die aktuelle Position
PDF_curveto Zeichnet eine Kurve
PDF_lineto Zeichnet eine Linie
PDF_circle Zeichnet einen Kreis
PDF_arc Zeichnet einen Kreisbogen
PDF_rect Zeichnet ein Rechteck
PDF_closepath Schließt einen Pfad
PDF_stroke Zeichnet eine Linie entlang eines Pfades
PDF_closepath_stroke Schließt einen Pfad und zeichnet eine Linie entlang des Pfades
PDF_fill Füllt den aktuellen Pfad
PDF_fill_stroke Füllt den aktuellen Pfad und zeichnet eine Linie entlang des Pfades
PDF_closepath_fill_stroke Schließt, füllt und zeichnet eine Linie entlang des Pfades
PDF_endpath Beendet den aktuellen Pfad
PDF_clip Begrenzt alle Zeichenoperation auf den aktuellen Pfad
PDF_setgray_fill Setzt die Füllfarbe auf einen Grauwert
PDF_setgray_stroke Setzt die Zeichenfarbe auf einen Grauwert
PDF_setgray Setzt die Zeichen- und Füllfarbe auf einen Grauwert
PDF_setrgbcolor_fill Setzt die Füllfarbe auf einen Farbwert
PDF_setrgbcolor_stroke Setzt die Zeichenfarbe auf einen Farbwert
PDF_setrgbcolor Setzt die Zeichen- und Füllfarbe auf einen Farbwert
PDF_add_outline Fügt Lesemarke zur aktuellen Seite hinzu
PDF_set_transition Setzt den Übergang zur nächsten Seite
PDF_set_duration Setzt die Zeitdauer bis zur nächsten Seite
PDF_open_gif Öffnet ein GIF-Bild
PDF_open_memory_image Liest ein Bild, das mit PHP erzeugt wurde
PDF_open_jpeg Öffnet ein JPEG-Bild
PDF_close_image Schließt ein Bild
PDF_place_image Plaziert ein Bild auf der Seite
PDF_put_image Speichert ein Bild im PDF-Dokument für späteren Gebrauch
PDF_execute_image Plaziert ein gespeichertes Bild auf der Seite
pdf_add_annotation Fügt eine Anmerkung hinzu

XLV. Perl-compatible Regular Expression 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.

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]/

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.

Inhaltsverzeichnis

XLVI. PHP OPtionen und Informationen

Inhaltsverzeichnis

XLVII. POSIX functions

This module contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means. POSIX.1 for example defined the open(), read(), write() and close() functions, too, which traditionally have been part of PHP3 for a long time. Some more system specific functions have not been available before, though, and this module tries to remedy this by providing easy access to these functions.

Inhaltsverzeichnis

posix_getpwnam Return info about a user by username

posix_getpwuid Return info about a user by user id

posix_getrlimit Return info about system ressource limits

XLVIII. 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.

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");	OK
postmaster -i &	pg_connect("", "", "", "", "dbname");	OK
postmaster &	pg_connect("localhost", "", "", "", "dbname");	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("localhost", "", "", "", "dbname");	OK

Eine Verbindung läßt sich auch mit folgendem Befehl herstellen: **$conn = pg_Connect("host=localhost port=5432 dbname=chris");**

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** und **end**. Wenn die Transaktion fehlschlägt, sollte sie mit **abort** und **rollback** geschlossen werden.

Beispiel 1 Grosse Objekte benutzen

```
 1
 2 <?php
 3 $database = pg_Connect ("", "", "", "", "jacarta");
 4 pg_exec ($database, "begin");
 5     $oid = pg_locreate ($database);
 6     echo ("$oid\n");
 7     $handle = pg_loopen ($database, $oid, "w");
 8     echo ("$handle\n");
 9     pg_lowrite ($handle, "gaga");
10     pg_loclose ($handle);
11 pg_exec ($database, "commit")
12 pg_exec ($database, "end")
13 ?>
14
```

Inhaltsverzeichnis

pg_lounlink Ein grosses Objekt löschen

pg_lowrite Schreibt in ein grosses Objekt

pg_NumFields Gibt die Anzahl der Felder aus

pg_NumRows Gibt die Anzahl der Zeilen aus

pg_Options Liefert Verbindungsoptionen

pg_pConnect Stellt eine persistente Datenbankverbindung her

pg_Port Gibt die Portnummer aus

pg_Result Liefert Werte eines bezeichneten Ergebnisses

pg_tty Gibt den tty-Namen aus

XLIX. Program Execution functions

Inhaltsverzeichnis

L. GNU Recode functions

This module contains an interface to the GNU Recode library, version 3.5. To be able to use the functions defined in this module you must compile you PHP interpreter using the --with-recode option. In order to do so, you must have GNU Recode 3.5 or higher installed on your system.

The GNU Recode library converts files between various coded character sets and surface encodings. When this cannot be achieved exactly, it may get rid of the offending characters or fall back on approximations. The library recognises or produces nearly 150 different character sets and is able to convert files between almost any pair. Most RFC 1345 character sets are supported.

Inhaltsverzeichnis

LI. Regular expression functions

Regular expressions are used for complex string manipulation in PHP. The functions that support regular expressions are:

- **ereg()**
- **ereg_replace()**
- **eregi()**
- **eregi_replace()**
- **split()**

These functions all take a regular expression string as their first argument. PHP uses the POSIX extended regular expressions as defined by POSIX 1003.2. For a full description of POSIX regular expressions see the regex man pages included in the regex directory in the PHP distribution. It's in manpage format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

Beispiel 1 Regular expression examples

```
 1
 2 ereg("abc",$string);
 3 /* Returns true if "abc"
 4    is found anywhere in $string. */
 5
 6 ereg("^abc",$string);
 7 /* Returns true if "abc"
 8    is found at the beginning of $string. */
 9
10 ereg("abc$",$string);
11 /* Returns true if "abc"
12    is found at the end of $string. */
13
14 eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
15 /* Returns true if client browser
16    is Netscape 2, 3 or MSIE 3. */
17
18 ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)",
19      $string,$regs);
20 /* Places three space separated words
21    into $regs[1], $regs[2] and $regs[3]. */
22
23 $string = ereg_replace("^","<BR>",$string);
24 /* Put a <BR> tag at the beginning of $string. */
```

```
25
26 $string = ereg_replace("$","<BR>",$string);
27 /* Put a <BR> tag at the end of $string. */
28
29 $string = ereg_replace("\n","",$string);
30 /* Get rid of any carriage return
31    characters in $string. */
32
```

Inhaltsverzeichnis

LII. Semaphore and Shared Memory Functions

This module provides semaphore functions using System V semaphores. Semaphores may be used to provide exclusive access to resources on the current machine, or to limit the number of processes that may simultaneously use a resource.

This module provides also shared memory functions using System V shared memory. Shared memory may be used to provide access to global variables. Different httpd-daemons and even other programs (such as Perl, C, ...) are able to access this data to provide a global data-exchange. Remember, that shared memory is NOT safe against simultaneous access. Use semaphores for synchronization.

Tabelle 1 Limits of Shared Memory by the Unix OS

SHMMAX	max size of shared memory, normally 131072 bytes
SHMMIN	minimum size of shared memory, normally 1 byte
SHMMNI	max amount of shared memory segments, normally 100
SHMSEG	max amount of shared memory per process, normally 6

Inhaltsverzeichnis

LIII. Session handling functions

Session support in PHP consists of a way to preserve certain data across subsequent accesses. This enables you to build more customized applications and increase the appeal of your web site.

If you are familiar with the session management of PHPLIB, you will notice that some concepts are similar to PHP's session support.

A visitor accessing your web site is assigned an unique id, the so-called session id. This is either stored in a cookie on the user side or is propagated in the URL.

The session support allows you to register arbitrary numbers of variables to be preserved across requests. When a visitor accesses your site, PHP will check automatically (if session.auto_start is set to 1) or on your request (explicitly through **session_start()** or implicitly through **session_register()**) whether a specific session id has been sent with the request. If this is the case, the prior saved environment is recreated.

All registered variables are serialized after the request finishes. Registered variables which are undefined are marked as being not defined. On subsequent accesses, these are not defined by the session module unless the user defines them later.

`track_vars` and `gpc_globals` configuration settings influence how the session variables get restored. If `track_vars` is enabled, then the restored session variables will be available in the global associative array $HTTP_STATE_VARS. If `gpc_globals` is enabled, then the session variables will be restored to corresponding global variables. If both of these settings are enabled, then the globals variables and the $HTTP_STATE_VARS entries will reference the same value.

There are two methods to propagate a session id:

- Cookies
- URL parameter

The session module supports both methods. Cookies are optimal, but since they are not reliable (clients are not bound to accept them), we cannot rely on them. The second method embeds the session id directly into URLs.

PHP is capable of doing this transparently when compiled with `--enable-trans-sid`. If you enable this option, relative URIs will be changed to contain the session id automatically. Alternatively, you can use the constant `SID` which is defined, if the client did not send the appropiate cookie. `SID` is either of the form `session_name=session_id` or is an empty string.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

Beispiel 1 Counting the number of hits of a single user

```
 1
 2 <?php
 3 session_register("count");
 4 $count++;
 5 ?>
 6
 7 Hello visitor, you have seen this page <? echo $count; ?> times.<p>
 8
 9 <php?
10 # the <?=SID?> is necessary to preserve the session id
11 # in the case that the user has disabled cookies
12 ?>
13
14 To continue, <A HREF="nextpage.php?<?=SID?>">click here</A>
15
```

To implement database storage you need PHP code and a user level function **session_set_save_handler()**. You would have to extend the following functions to cover MySQL or another database.

Beispiel 2 Usage of session_set_save_handler()

```
 1
 2 <?php
 3
 4 function open ($save_path, $session_name) {
 5     echo "open ($save_path, $session_name)\n";
 6     return true;
 7 }
 8
 9 function close() {
10     echo "close\n";
11     return true;
12 }
13
14 function read ($key) {
15     echo "write ($key, $val)\n";
16     return "foo|i:1;";
17 }
18
19 function write ($key, $val) {
20     echo "write ($key, $val)\n";
21     return true;
22 }
23
24 function destroy ($key) {
25     return true;
26 }
27
28 function gc ($maxlifetime) {
29     return true;
30 }
31
32 session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");
33
34 session_start();
35
36 $foo++;
37
38 ?>
39
```

Will produce this results:

```
 1
 2 $ ./php save_handler.php
 3 Content-Type: text/html
 4 Set-cookie: PHPSESSID=f08b925af0ecb52bdd2de97d95cdbe6b
 5
 6 open (/tmp, PHPSESSID)
 7 read (f08b925af0ecb52bdd2de97d95cdbe6b)
 8 write (f08b925af0ecb52bdd2de97d95cdbe6b, foo|i:2;)
 9 close
10
```

The `<?=SID?>` is not necessary, if `--enable-trans-sid` was used to compile PHP.

The session management system supports a number of configuration options which you can place in your php.ini file. We

will give a short overview.

- `session.save_handler` defines the name of the handler which is used for storing and retrieving data associated with a session. Defaults to `files`.
- `session.save_path` defines the argument which is passed to the save handler. If you choose the default files handler, this is the path where the files are created. Defaults to `/tmp`.
- `session.name` specifies the name of the session which is used as cookie name. It should only contain alphanumeric characters. Defaults to `PHPSESSID`.
- `session.auto_start` specifies whether the session module start a session automatically on request startup. Defaults to `0` (disabled).
- `session.lifetime` specifies the lifetime of the cookie in seconds which is sent to the browser. The value 0 means "until the browser is closed." Defaults to `0`.
- `session.serialize_handler` defines the name of the handler which is used to serialize/deserialize data. Currently, a PHP internal format (name `php`) and WDDX is supported (name `wddx`). WDDX is only available, if PHP is compiled with WDDX support. Defaults to `php`.
- `session.gc_probability` specifies the probability that the gc (garbage collection) routine is started on each request in percent. Defaults to `1`.
- `session.gc_maxlifetime` specifies the number of seconds after which data will be seen as 'garbage' and cleaned up.
- `session.referer_check` determines whether session ids referred to by external sites will be eliminated. If session ids are propagated using the URL method, users not knowing about the impact might publish session ids. This can lead to security problems which this check tries to defeat. Defaults to `0`.
- `session.entropy_file` gives a path to an external resource (file) which will be used as an additional entropy source in the session id creation process. Examples are `/dev/random` or `/dev/urandom` which are available on many Unix systems.
- `session.entropy_length` specifies the number of bytes which will be read from the file specified above. Defaults to `0` (disabled).
- `session.use_cookies` specifies whether the module will use cookies to store the session id on the client side. Defaults to `1` (enabled).

Anmerkung: Session handling was added in PHP 4.0.

Inhaltsverzeichnis

LIV. SNMP functions

In order to use the SNMP functions on Unix you need to install the UCD SNMP package. On Windows these functions are only available on NT and not on Win95/98.

Important: In order to use the UCD SNMP package, you need to define NO_ZEROLENGTH_COMMUNITY to 1 before compiling it. After configuring UCD SNMP, edit config.h and search for NO_ZEROLENGTH_COMMUNITY. Uncomment the #define line. It should look like this afterwards:

```
1
2 #define NO_ZEROLENGTH_COMMUNITY 1
3
```

If you see strange segmentation faults in combination with SNMP commands, you did not follow the above instructions. If you do not want to recompile UCD SNMP, you can compile PHP with the --enable-ucd-snmp-hack switch which will work around the misfeature.

Inhaltsverzeichnis

LV. String functions

These functions all manipulate strings in various ways. Some more specialized sections can be found in the regular expression and URL handling sections.

Inhaltsverzeichnis

QuoteMeta quote meta characters
rawurldecode Decode URL-encoded strings
rawurlencode URL-encode according to RFC1738.
setlocale Set locale information.
similar_text Calculate the similarity between two strings.
soundex Calculate the soundex key of a string
sprintf Return a formatted string
strcasecmp Binary safe case-insensitive string comparison.
strchr Find the first occurrence of a character.
strcmp Binary safe string comparison
strcspn Find length of initial segment not matching mask.
strip_tags Strip HTML and PHP tags from a string
StripCSlashes un-quote string quoted with addcslashes
StripSlashes Un-quote string quoted with addslashes
stristr Case-insensitive **strstr()**.
strlen Get string length.
strpos Find position of first occurrence of a string.
strrchr Find the last occurrence of a character in a string.
str_repeat Repeat a string.
strrev Reverse a string.
strrpos Find position of last occurrence of a char in a string.
strspn Find length of initial segment matching mask.
strstr Find first occurrence of a string.
strtok Tokenize string
strtolower Make a string lowercase.
strtoupper Make a string uppercase.
str_replace Replace all occurrences of needle in haystack with str.
strtr Translate certain characters.
substr Return part of a string
substr_replace Replace text within a portion of a string.
trim Strip whitespace from the beginning and end of a string.
ucfirst Make a string's first character uppercase.
ucwords Uppercase the first character of each word in a string

---

LVI. Sybase functions

Inhaltsverzeichnis

LVII. URL functions

Inhaltsverzeichnis

LVIII. Variablen-Functions

Inhaltsverzeichnis

LIX. Vmailmgr functions

These functions require qmail and the vmailmgr package by Bruce Guenter.

For all functions, the following two variables are defined as: string vdomain the domain name of your virtual domain (vdomain.com) string basepwd the password of the 'real' user that holds the virtual users

Only up to 8 characters are recognized in passwords for virtual users

Return status for all functions matches response in response.h

O ok
1 bad
2 error
3 error connecting

Known problems: **vm_deluser()** does not delete the user directory as it should. **vm_addalias()** currently does not work correctly.

```
1
2 <?php
3 dl("php3_vmailmgr.so"); //load the shared library
4 $vdomain="vdomain.com";
5 $basepwd="password";
6 ?>
7
```

Inhaltsverzeichnis

LX. WDDX functions

These functions are intended for work with WDDX.

Note that all the functions that serialize variables use the first element of an array to determine whether the array is to be serialized into an array or structure. If the first element has string key, then it is serialized into a structure, otherwise, into an array.

Beispiel 1 Serializing a single value

```
 1
 2 <?php
 3 print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
 4 ?>
 5
```

This example will produce:

```
 1
 2 <wddxPacket version='0.9'><header comment='PHP packet'/><data>
 3 <string>PHP to WDDX packet example</string></data></wddxPacket>
 4
```

Beispiel 2 Using incremental packets

```
 1
 2 <?php
 3 $pi = 3.1415926;
 4 $packet_id = wddx_packet_start("PHP");
 5 wddx_add_vars($packet_id, "pi");
 6
 7 /* Suppose $cities came from database */
 8 $cities = array("Austin", "Novato", "Seattle");
 9 wddx_add_vars($packet_id, "cities");
10
11 $packet = wddx_packet_end($packet_id);
12 print $packet;
13 ?>
14
```

This example will produce:

```
 1
 2 <wddxPacket version='0.9'><header comment='PHP'/><data><struct>
 3 <var name='pi'><number>3.1415926</number></var><var name='cities'>
 4 <array length='3'><string>Austin</string><string>Novato</string>
 5 <string>Seattle</string></array></var></struct></data></wddxPacket>
 6
```

Inhaltsverzeichnis

wddx_packet_start Starts a new WDDX packet with structure inside it

wddx_packet_end Ends a WDDX packet with the specified ID

wddx_add_vars Ends a WDDX packet with the specified ID

wddx_deserialize Deserializes a WDDX packet

LXI. 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:

```
1
2 libexpat.a: $(OBJS)
3     ar -rc $@ $(OBJS)
4     ranlib $@
5
```

A source RPM package of expat can be found at http://www.guardian.no/~ssb/phpxml.html.

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
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 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

```
 1
 2 $file = "data.xml";
 3 $depth = array();
 4
 5 function startElement($parser, $name, $attrs) {
 6     global $depth;
 7     for ($i = 0; $i < $depth[$parser]; $i++) {
 8         print "  ";
 9     }
10     print "$name\n";
11     $depth[$parser]++;
12 }
13
14 function endElement($parser, $name) {
15     global $depth;
16     $depth[$parser]--;
17 }
18
19 $xml_parser = xml_parser_create();
20 xml_set_element_handler($xml_parser, "startElement", "endElement");
21 if (!($fp = fopen($file, "r"))) {
22     die("could not open XML input");
23 }
24
25 while ($data = fread($fp, 4096)) {
26     if (!xml_parse($xml_parser, $data, feof($fp))) {
27         die(sprintf("XML error: %s at line %d",
28                     xml_error_string(xml_get_error_code($xml_parser)),
29                     xml_get_current_line_number($xml_parser)));
30     }
31 }
32 xml_parser_free($xml_parser);
33
```

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.

```
 1
 2 $file = "data.xml";
 3 $map_array = array(
 4     "BOLD"     => "B",
 5     "EMPHASIS" => "I",
 6     "LITERAL"  => "TT"
 7 );
 8
 9 function startElement($parser, $name, $attrs) {
10     global $map_array;
11     if ($htmltag = $map_array[$name]) {
12         print "<$htmltag>";
13     }
14 }
15
16 function endElement($parser, $name) {
17     global $map_array;
18     if ($htmltag = $map_array[$name]) {
19         print "</$htmltag>";
20     }
21 }
22
23 function characterData($parser, $data) {
24     print $data;
25 }
26
27 $xml_parser = xml_parser_create();
28 // use case-folding so we are sure to find the tag in $map_array
29 xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
30 xml_set_element_handler($xml_parser, "startElement", "endElement");
31 xml_set_character_data_handler($xml_parser, "characterData");
32 if (!($fp = fopen($file, "r"))) {
33     die("could not open XML input");
34 }
35
36 while ($data = fread($fp, 4096)) {
37     if (!xml_parse($xml_parser, $data, feof($fp))) {
38         die(sprintf("XML error: %s at line %d",
39                     xml_error_string(xml_get_error_code($xml_parser)),
40                     xml_get_current_line_number($xml_parser)));
41     }
42 }
43 xml_parser_free($xml_parser);
44
```

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

```
 1
 2 $file = "xmltest.xml";
 3
 4 function trustedFile($file) {
 5     // only trust local files owned by ourselves
 6     if (!eregi("^([a-z]+)://", $file)
 7         && fileowner($file) == getmyuid()) {
 8             return true;
 9     }
10     return false;
11 }
12
13 function startElement($parser, $name, $attribs) {
14     print "&lt;<font color=\"#0000cc\">$name</font>";
15     if (sizeof($attribs)) {
16         while (list($k, $v) = each($attribs)) {
17             print " <font color=\"#009900\">$k</font>=\"<font
18                    color=\"#990000\">$v</font>\"";
19         }
20     }
21     print "&gt;";
22 }
23
24 function endElement($parser, $name) {
25     print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
26 }
27
28 function characterData($parser, $data) {
29     print "<b>$data</b>";
30 }
31
32 function PIHandler($parser, $target, $data) {
33     switch (strtolower($target)) {
34         case "php":
35             global $parser_file;
36             // If the parsed document is "trusted", we say it is safe
37             // to execute PHP code inside it.  If not, display the code
38             // instead.
39             if (trustedFile($parser_file[$parser])) {
40                 eval($data);
41             } else {
42                 printf("Untrusted PHP code: <i>%s</i>",
43                        htmlspecialchars($data));
44             }
45             break;
46     }
47 }
48
49 function defaultHandler($parser, $data) {
50     if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
51         printf('<font color="#aa00aa">%s</font>',
52                htmlspecialchars($data));
53     } else {
54         printf('<font size="-1">%s</font>',
55                htmlspecialchars($data));
56     }
57 }
58
59 function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
60                                   $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

```
 1
 2 <?xml version='1.0'?>
 3 <!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
 4 <!ENTITY plainEntity "FOO entity">
 5 <!ENTITY systemEntity SYSTEM "xmltest2.xml">
 6 ]>
 7 <chapter>
 8  <TITLE>Title &plainEntity;</TITLE>
 9  <para>
10   <informaltable>
11    <tgroup cols="3">
12     <tbody>
13      <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
14      <row><entry>a2</entry><entry>c2</entry></row>
15      <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
16     </tbody>
17    </tgroup>
18   </informaltable>
19  </para>
20  &systemEntity;
21  <sect1 id="about">
22   <title>About this Document</title>
23   <para>
24    <!-- this is a comment -->
25    <?php print 'Hi!  This is PHP version '.phpversion(); ?>
26   </para>
27  </sect1>
28 </chapter>
29
```

This file is included from `xmltest.xml`:

Beispiel 5 xmltest2.xml

```
 1
 2 <?xml version="1.0"?>
 3 <!DOCTYPE foo [
 4 <!ENTITY testEnt "test entity">
 5 ]>
 6 <foo>
 7    <element attrib="value"/>
 8    &testEnt;
 9    <?php print "This is some more PHP code being executed."; ?>
10 </foo>
11
```

Inhaltsverzeichnis

xml_error_string get XML parser error string

xml_get_current_line_number get current line number for an XML parser

xml_get_current_column_number Get current column number for an XML parser

xml_get_current_byte_index get current byte index for an XML parser

xml_parser_free Free an XML parser

xml_parser_set_option set options in an XML parser

xml_parser_get_option get options from an XML parser

utf8_decode Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1.

utf8_encode encodes an ISO-8859-1 string to UTF-8

V Anhang

Inhaltsverzeichnis

Anhang A Migrating from PHP/FI 2.0 to PHP 3.0

Inhaltsverzeichnis

About the incompatbilities in 3.0

PHP 3.0 is rewritten from the ground up. It has a proper parser that is much more robust and consistent than 2.0's. 3.0 is also significantly faster, and uses less memory. However, some of these improvements have not been possible without compatibility changes, both in syntax and functionality.

In addition, PHP's developers have tried to clean up both PHP's syntax and semantics in version 3.0, and this has also caused some incompatibilities. In the long run, we believe that these changes are for the better.

This chapter will try to guide you through the incompatibilities you might run into when going from PHP/FI 2.0 to PHP 3.0 and help you resolve them. New features are not mentioned here unless necessary.

A conversion program that can automatically convert your old PHP/FI 2.0 scripts exists. It can be found in the `convertor` subdirectory of the PHP 3.0 distribution. This program only catches the syntax changes though, so you should read this chapter carefully anyway.

Anhang B PHP development

Inhaltsverzeichnis

Adding functions to PHP3

Function Prototype

All functions look like this:

```
1
2 void php3_foo(INTERNAL_FUNCTION_PARAMETERS) {
3
4 }
5
```

Even if your function doesn't take any arguments, this is how it is called.

Function Arguments

Arguments are always of type pval. This type contains a union which has the actual type of the argument. So, if your function takes two arguments, you would do something like the following at the top of your function:

Beispiel B-1 Fetching function arguments

```
1
2 pval *arg1, *arg2;
3 if (ARG_COUNT(ht) != 2 || getParameters(ht,2,&arg1,&arg2)==FAILURE) {
4    WRONG_PARAM_COUNT;
5 }
6
```

NOTE: Arguments can be passed either by value or by reference. In both cases you will need to pass &(pval *) to getParameters. If you want to check if the n'th parameter was sent to you by reference or not, you can use the function, ParameterPassedByReference(ht,n). It will return either 1 or 0.

When you change any of the passed parameters, whether they are sent by reference or by value, you can either start over with the parameter by calling pval_destructor on it, or if it's an ARRAY you want to add to, you can use functions similar to the ones in internal_functions.h which manipulate return_value as an ARRAY.

Also if you change a parameter to IS_STRING make sure you first assign the new estrdup()'ed string and the string length, and only later change the type to IS_STRING. If you change the string of a parameter which already IS_STRING or IS_ARRAY you should run pval_destructor on it first.

Variable Function Arguments

A function can take a variable number of arguments. If your function can take either 2 or 3 arguments, use the following:

Beispiel B-2 Variable function arguments

```
  1
  2 pval *arg1, *arg2, *arg3;
  3 int arg_count = ARG_COUNT(ht);
  4
  5 if (arg_count < 2 || arg_count > 3 ||
  6     getParameters(ht,arg_count,&arg1,&arg2,&arg3)==FAILURE) {
  7     WRONG_PARAM_COUNT;
  8 }
  9
```

Using the Function Arguments

The type of each argument is stored in the pval type field. This type can be any of the following:

Tabelle B-1 PHP Internal Types

IS_STRING	String
IS_DOUBLE	Double-precision floating point
IS_LONG	Long integer
IS_ARRAY	Array
IS_EMPTY	None
IS_USER_FUNCTION	??
IS_INTERNAL_FUNCTION	?? (if some of these cannot be passed to a function - delete)
IS_CLASS	??
IS_OBJECT	??

If you get an argument of one type and would like to use it as another, or if you just want to force the argument to be of a certain type, you can use one of the following conversion functions:

```
  1
  2 convert_to_long(arg1);
  3 convert_to_double(arg1);
  4 convert_to_string(arg1);
  5 convert_to_boolean_long(arg1); /* If the string is "" or "0" it becomes 0, 1
otherwise */
  6 convert_string_to_number(arg1);  /* Converts string to either LONG or DOUBLE
depending on string */
  7
```

These function all do in-place conversion. They do not return anything.

The actual argument is stored in a union; the members are:

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

Memory Management in Functions

Any memory needed by a function should be allocated with either emalloc() or estrdup(). These are memory handling abstraction functions that look and smell like the normal malloc() and strdup() functions. Memory should be freed with efree().

There are two kinds of memory in this program: memory which is returned to the parser in a variable, and memory which you need for temporary storage in your internal function. When you assign a string to a variable which is returned to the parser you need to make sure you first allocate the memory with either emalloc() or estrdup(). This memory should NEVER be freed by you, unless you later in the same function overwrite your original assignment (this kind of programming practice is not good though).

For any temporary/permanent memory you need in your functions/library you should use the three emalloc(), estrdup(), and efree() functions. They behave EXACTLY like their counterpart functions. Anything you emalloc() or estrdup() you have to efree() at some point or another, unless it's supposed to stick around until the end of the program; otherwise, there will be a memory leak. The meaning of "the functions behave exactly like their counterparts" is: if you efree() something which was not emalloc()'ed nor estrdup()'ed you might get a segmentation fault. So please take care and free all of your wasted memory.

If you compile with "-DDEBUG", PHP3 will print out a list of all memory that was allocated using emalloc() and estrdup() but never freed with efree() when it is done running the specified script.

Setting Variables in the Symbol Table

A number of macros are available which make it easier to set a variable in the symbol table:

- SET_VAR_STRING(name,value) [1]
- SET_VAR_DOUBLE(name,value)
- SET_VAR_LONG(name,value)

[1]

Symbol tables in PHP 3.0 are implemented as hash tables. At any given time, &symbol_table is a pointer to the 'main' symbol table, and active_symbol_table points to the currently active symbol table (these may be identical like in startup, or different, if you're inside a function).

The following examples use 'active_symbol_table'. You should replace it with &symbol_table if you specifically want to work with the 'main' symbol table. Also, the same functions may be applied to arrays, as explained below.

Beispiel B-3 Checking whether $foo exists in a symbol table

```
1
2 if (hash_exists(active_symbol_table,"foo",sizeof("foo"))) { exists... }
3 else { doesn't exist }
4
```

Beispiel B-4 Finding a variable's size in a symbol table

```
1
2 hash_find(active_symbol_table,"foo",sizeof("foo"),&pvalue);
3 check(pvalue.type);
4
```

Arrays in PHP 3.0 are implemented using the same hashtables as symbol tables. This means the two above functions can also be used to check variables inside arrays.

If you want to define a new array in a symbol table, you should do the following.

First, you may want to check whether it exists and abort appropiately, using hash_exists() or hash_find().

Next, initialize the array:

Beispiel B-5 Initializing a new array

```
1
2 pval arr;
3
4 if (array_init(&arr) == FAILURE) { failed... };
5 hash_update(active_symbol_table,"foo",sizeof("foo"),&arr,sizeof(pval),NULL);
6
```

This code declares a new array, named $foo, in the active symbol table. This array is empty.

Here's how to add new entries to it:

Beispiel B-6 Adding entries to a new array

```
 1
 2 pval entry;
 3
 4 entry.type = IS_LONG;
 5 entry.value.lval = 5;
 6
 7 /* defines $foo["bar"] = 5 */
 8 hash_update(arr.value.ht,"bar",sizeof("bar"),&entry,sizeof(pval),NULL);
 9
10 /* defines $foo[7] = 5 */
11 hash_index_update(arr.value.ht,7,&entry,sizeof(pval),NULL);
12
13 /* defines the next free place in $foo[],
14  * $foo[8], to be 5 (works like php2)
15  */
16 hash_next_index_insert(arr.value.ht,&entry,sizeof(pval),NULL);
17
```

If you'd like to modify a value that you inserted to a hash, you must first retrieve it from the hash. To prevent that overhead, you can supply a pval ** to the hash add function, and it'll be updated with the pval * address of the inserted element inside the hash. If that value is NULL (like in all of the above examples) - that parameter is ignored.

hash_next_index_insert() uses more or less the same logic as "$foo[] = bar;" in PHP 2.0.

If you are building an array to return from a function, you can initialize the array just like above by doing:

```
 1
 2 if (array_init(return_value) == FAILURE) { failed...; }
 3
```

...and then adding values with the helper functions:

```
 1
 2 add_next_index_long(return_value,long_value);
 3 add_next_index_double(return_value,double_value);
 4 add_next_index_string(return_value,estrdup(string_value));
 5
```

Of course, if the adding isn't done right after the array initialization, you'd probably have to look for the array first:

```
 1
 2 pval *arr;
 3
 4 if (hash_find(active_symbol_table,"foo",sizeof("foo"),(void **)&arr)==FAILURE) {
can't find... }
 5 else { use arr->value.ht... }
 6
```

Note that hash_find receives a pointer to a pval pointer, and not a pval pointer.

Just about any hash function returns SUCCESS or FAILURE (except for hash_exists(), which returns a boolean truth value).

Returning simple values

A number of macros are available to make returning values from a function easier.

The RETURN_* macros all set the return value and return from the function:

- RETURN
- RETURN_FALSE
- RETURN_TRUE
- RETURN_LONG(l)

- RETURN_STRING(s,dup) If dup is true, duplicates the string
- RETURN_STRINGL(s,l,dup) Return string (s) specifying length (l).
- RETURN_DOUBLE(d)

The RETVAL_* macros set the return value, but do not return.

- RETVAL_FALSE
- RETVAL_TRUE
- RETVAL_LONG(l)
- RETVAL_STRING(s,dup) If dup is true, duplicates the string
- RETVAL_STRINGL(s,l,dup) Return string (s) specifying length (l).
- RETVAL_DOUBLE(d)

The string macros above will all estrdup() the passed 's' argument, so you can safely free the argument after calling the macro, or alternatively use statically allocated memory.

If your function returns boolean success/error responses, always use RETURN_TRUE and RETURN_FALSE respectively.

Returning complex values

Your function can also return a complex data type such as an object or an array.

Returning an object:

1. Call object_init(return_value).
2. Fill it up with values. The functions available for this purpose are listed below.
3. Possibly, register functions for this object. In order to obtain values from the object, the function would have to fetch "this" from the active_symbol_table. Its type should be IS_OBJECT, and it's basically a regular hash table (i.e., you can use regular hash functions on .value.ht). The actual registration of the function can be done using:

```
1
2 add_method( return_value, function_name, function_ptr );
3
```

The functions used to populate an object are:

- add_property_long(return_value, property_name, l) - Add a property named 'property_name', of type long, equal to 'l'
- add_property_double(return_value, property_name, d) - Same, only adds a double
- add_property_string(return_value, property_name, str) - Same, only adds a string
- add_property_stringl(return_value, property_name, str, l) - Same, only adds a string of length 'l'

Returning an array:

1. Call array_init(return_value).
2. Fill it up with values. The functions available for this purpose are listed below.

The functions used to populate an array are:

- add_assoc_long(return_value,key,l) - add associative entry with key 'key' and long value 'l'
- add_assoc_double(return_value,key,d)
- add_assoc_string(return_value,key,str,duplicate)
- add_assoc_stringl(return_value,key,str,length,duplicate) specify the string length
- add_index_long(return_value,index,l) - add entry in index 'index' with long value 'l'
- add_index_double(return_value,index,d)

- add_index_string(return_value,index,str)
- add_index_stringl(return_value,index,str,length) - specify the string length
- add_next_index_long(return_value,l) - add an array entry in the next free offset with long value 'l'
- add_next_index_double(return_value,d)
- add_next_index_string(return_value,str)
- add_next_index_stringl(return_value,str,length) - specify the string length

Using the resource list

PHP 3.0 has a standard way of dealing with various types of resources. This replaces all of the local linked lists in PHP 2.0.

Available functions:

- php3_list_insert(ptr, type) - returns the 'id' of the newly inserted resource
- php3_list_delete(id) - delete the resource with the specified id
- php3_list_find(id,*type) - returns the pointer of the resource with the specified id, updates 'type' to the resource's type

Typically, these functions are used for SQL drivers but they can be used for anything else; for instance, maintaining file descriptors.

Typical list code would look like this:

Beispiel B-7 Adding a new resource

```
 1
 2 RESOURCE *resource;
 3
 4 /* ...allocate memory for resource and acquire resource... */
 5 /* add a new resource to the list */
 6 return_value->value.lval = php3_list_insert((void *) resource, LE_RESOURCE_TYPE);
 7 return_value->type = IS_LONG;
 8
```

Beispiel B-8 Using an existing resource

```
 1
 2 pval *resource_id;
 3 RESOURCE *resource;
 4 int type;
 5
 6 convert_to_long(resource_id);
 7 resource = php3_list_find(resource_id->value.lval, &type);
 8 if (type != LE_RESOURCE_TYPE) {
 9     php3_error(E_WARNING,"resource index %d has the wrong
type",resource_id->value.lval);
10     RETURN_FALSE;
11 }
12 /* ...use resource... */
13
```

Beispiel B-9 Deleting an existing resource

```
 1
 2 pval *resource_id;
 3 RESOURCE *resource;
 4 int type;
 5
 6 convert_to_long(resource_id);
 7 php3_list_delete(resource_id->value.lval);
 8
```

The resource types should be registered in php3_list.h, in enum list_entry_type. In addition, one should add shutdown code for

any new resource type defined, in list.c's list_entry_destructor() (even if you don't have anything to do on shutdown, you must add an empty case).

Using the persistent resource table

PHP 3.0 has a standard way of storing persistent resources (i.e., resources that are kept in between hits). The first module to use this feature was the MySQL module, and mSQL followed it, so one can get the general impression of how a persistent resource should be used by reading mysql.c. The functions you should look at are:

php3_mysql_do_connect
php3_mysql_connect()
php3_mysql_pconnect()

The general idea of persistence modules is this:

1. Code all of your module to work with the regular resource list mentioned in section (9).
2. Code extra connect functions that check if the resource already exists in the persistent resource list. If it does, register it as in the regular resource list as a pointer to the persistent resource list (because of 1., the rest of the code should work immediately). If it doesn't, then create it, add it to the persistent resource list AND add a pointer to it from the regular resource list, so all of the code would work since it's in the regular resource list, but on the next connect, the resource would be found in the persistent resource list and be used without having to recreate it. You should register these resources with a different type (e.g. LE_MYSQL_LINK for non-persistent link and LE_MYSQL_PLINK for a persistent link).

If you read mysql.c, you'll notice that except for the more complex connect function, nothing in the rest of the module has to be changed.

The very same interface exists for the regular resource list and the persistent resource list, only 'list' is replaced with 'plist':

- php3_plist_insert(ptr, type) - returns the 'id' of the newly inserted resource
- php3_plist_delete(id) - delete the resource with the specified id
- php3_plist_find(id,*type) - returns the pointer of the resource with the specified id, updates 'type' to the resource's type

However, it's more than likely that these functions would prove to be useless for you when trying to implement a persistent module. Typically, one would want to use the fact that the persistent resource list is really a hash table. For instance, in the MySQL/mSQL modules, when there's a pconnect() call (persistent connect), the function builds a string out of the host/user/passwd that were passed to the function, and hashes the SQL link with this string as a key. The next time someone calls a pconnect() with the same host/user/passwd, the same key would be generated, and the function would find the SQL link in the persistent list.

Until further documented, you should look at mysql.c or msql.c to see how one should use the plist's hash table abilities.

One important thing to note: resources going into the persistent resource list must *NOT* be allocated with PHP's memory manager, i.e., they should NOT be created with emalloc(), estrdup(), etc. Rather, one should use the regular malloc(), strdup(), etc. The reason for this is simple - at the end of the request (end of the hit), every memory chunk that was allocated using PHP's memory manager is deleted. Since the persistent list isn't supposed to be erased at the end of a request, one mustn't use PHP's memory manager for allocating resources that go to it.

When you register a resource that's going to be in the persistent list, you should add destructors to it both in the non-persistent list and in the persistent list. The destructor in the non-persistent list destructor shouldn't do anything. The one in the persistent list destructor should properly free any resources obtained by that type (e.g. memory, SQL links, etc). Just like with the non-persistent resources, you *MUST* add destructors for every resource, even it requires no destructotion and the destructor would be empty. Remember, since emalloc() and friends aren't to be used in conjunction with the persistent list, you mustn't use efree() here either.

Adding runtime configuration directives

Many of the features of PHP3 can be configured at runtime. These configuration directives can appear in either the designated php3.ini file, or in the case of the Apache module version in the Apache .conf files. The advantage of having them in the Apache .conf files is that they can be configured on a per-directory basis. This means that one directory may have a certain safemodeexecdir for example, while another directory may have another. This configuration granularity is especially handy when a server supports multiple virtual hosts.

The steps required to add a new directive:

1. Add directive to php3_ini_structure struct in mod_php3.h.
2. In main.c, edit the php3_module_startup function and add the appropriate cfg_get_string() or cfg_get_long() call.
3. Add the directive, restrictions and a comment to the php3_commands structure in mod_php3.c. Note the restrictions part. RSRC_CONF are directives that can only be present in the actual Apache .conf files. Any OR_OPTIONS directives can be present anywhere, include normal .htaccess files.
4. In either php3take1handler() or php3flaghandler() add the appropriate entry for your directive.
5. In the configuration section of the _php3_info() function in functions/info.c you need to add your new directive.
6. And last, you of course have to use your new directive somewhere. It will be addressable as php3_ini.directive.

Fußnoten

[1] Be careful here. The value part must be malloc'ed manually because the memory management code will try to free this pointer later. Do not pass statically allocated memory into a SET_VAR_STRING.

Anhang C The PHP Debugger

Inhaltsverzeichnis

Using the Debugger

PHP's internal debugger is useful for tracking down evasive bugs. The debugger works by connecting to a TCP port for every time PHP starts up. All error messages from that request will be sent to this TCP connection. This information is intended for "debugging server" that can run inside an IDE or programmable editor (such as Emacs).

How to set up the debugger:

1. Set up a TCP port for the debugger in the configuration file (debugger.port) and enable it (debugger.enabled).
2. Set up a TCP listener on that port somewhere (for example **socket -l -s 1400** on UNIX).
3. In your code, run "debugger_on(*`host`*)", where *`host`* is the IP number or name of the host running the TCP listener.

Now, all warnings, notices etc. will show up on that listener socket, *even if you them turned off with **error_reporting()***.