PHP3 Manual

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

Edited by Stig Sæther Bakken

¹ø¿ª : È²¿øÈñ (Home Page: http://w3.to/regina E-Mail: regina@officeware.medialab.co.kr)
(ÀÌ ±ÛÀº È²¿øÈñ °³ÀÎÀÌ ¹ø¿ªÇÑ ³»¿ëÀÔ´Ï´Ù. **±ÛÀÇ ¿À¿ª¿¡ µû¸¥ Å¥ÀÓÀ» ÁöÁö ¼Ê½À´Ï´Ù.** ±ÛÀÇ ÀßºøµÈ ºÎºÐÀº ¿¬¶ôÇØ ÁÖ½Ã¸é °¨»çÇÏ°Ú½À´Ï´Ù.)

Dedication

ÀÛ¼ºÀÏ : 1999-05-17

Release : 3.0.8

Â÷·Ê

¼ - ¹®

PHP Version 3.0Àº HTML¿¡ ³»ÅåµÇ¾î µ¿ÀÛÇÏ´Â ½ºÅ©¸³Æ® ¾ð¾îÀÌ´Ù.(HTML - embedded scripting language) PHP´Â C¿Í Java, PerlµîÀ¸·Î ºÎÅÍ ¸¹Àº ¹®Àå Çü½ÄÀ» º¹ô·Á¾²°í ÀÖ°í, ¸î°¡Áö´Â °íÀ¯Ç Ñ °Íµµ ÀÖ´Ù. ÀÌ ¾ð¾îÀÇ ¸ñÀûÀº À¥°³¹ßÀÚµéÀÌ µ¿ÀûÀÎ À¥¹®¼¸¦ ºü¸£°í ½±°Ô ÀÛ¼ºÇÒ ¼ö ÀÖµµ·Ï ÇÏ´Â °ÍÀÌ´Ù.

ÀÌ ¸Å´º¾ó¿¡ ´ëÇÏ¿©

¿ø·¡ µ¹® ¸Å´º¾óÀº DocBook DTDÀ» »ç¿ëÇÑ SGML·Î ÀÛ¼ºµÇ¾î ÀÖ°í, formattingÀ» À§ÇØ¼´Â DSSSL (Document Style and Semantics Specification Language)À» »ç¿ëÇÏ°í ÀÖ´Ù. ¶ÇÇÑ Tex¿Í RTF¹öÁ¯µµ ÀÖÀ¸³ª, ÇÑ±Û ¹®¼´Â ¿ªÀÚÀÇ ÆíÀÇ¸¦ À§ÇØ ÀÏ´Ü HTML¸¸À» »ç¿ëÇÏµµ·Ï ÇÏ°Ú´Ù.

I. Language Reference

Table of Contents

Chapter 1. PHP3ÀÇ ¼Ò°³ (An introduction to PHP3)

Table of Contents

PHP3¶õ ¹«¾ùÀÎ°¡?

PHP Version 3.0Àº server-side HTML-embedded scripting languageÀÔ´Ï´Ù.

PHP3´Â ¹«¾ù» ÇÒ ¼ö ÀÖ´Â°¡?

¾Æ¸¶ PHP3ÀÇ °¡Àå °-·ÂÇÏ°í °ü½É°¡´Â ºÎºÐÀº database¿ÍÀÇ ¿¬µ¿ºÎºÐÀÌ °ÍÀÌ´Ù. PHP3¸¦ »ç¿ëÇÏ¸é ¿©·¯ºÐÀº DatabaseÀÇ Data¸¦ »ç¿ëÇÑ Web page¸¦ ³î¶øµ·Ï °£´ÜÈ÷ ¸¸µé ¼ö ÀÖ´Ù. ´ÙÀ½¿¡ ³ª¿Â DB serverµéÀ» ÇöÀç »ç¿ëÇÒ ¼ö ÀÖ´Ù. :

Oracle	Adabas D
Sybase	FilePro
mSQL	Velocis
MySQL	Informix
Solid	dBase
ODBC	Unix dbm
PostgreSQL	

PHPÀÇ ¿ª»ç

PHP´Â 1994³â °¡À» Rasmus Lerdorf°¡ Ã³À½ °í¼ÈÇÏ¿´Ù. Ã³À½ ºñ°ø°³ÆÇÀº ±×ÀÇ È¨ÆäÀÌÁö¿¡ »ç¿ëµÇ¾ú°í, ¿Ü°Î¿¡ »ç¿ëÈ °ÍÀº 1995³â ÃÊ°ÎÍ »ç¿ëµÇ¾î Personal Home Page Tools¶ó°í °Ò¸®°Ô µÇ¾ú´Ù. ÀÌ°ÍÀº ¸î °³ÀÇ Æ¯°°ÇÑ ¸ÅÅ©·Î¸¦ »ç¿ëÇÒ ¼ö ÀÖ´Â ´Ü¼øÇÑ ÆÄ¼(¸í·É ÇØ¼®±â) ¿£Áø°ú ¹æ¸í·ÌÀÌ³ª Ä«¿îÅÍ °°ÀÌ È¨ÆäÀÌÁöÀÇ µÚ¿¡¼ °øÅëÀûÀ¸·Î »ç¿ëÇÒ ¼ö ÀÖ´Â ¸î °³ÀÇ °£´ÜÇÑ À¯Æ¿¸®Æ¼·Î ±¸¼º¾µÇ¾ú´Ù. ÀÌ ÆÄ¼°¡ 1995³â ÁßÀÝ¿¡ ÀçÀÛ¼º¾µÇ¾î PHP/FI Version 2¶ó°í ¸í¸íµÇ¾ú´Ù. FI´Â Rasmus°¡ ÀÛ¼ºÇÑ html Çü½ÄÀÇ µ¥ÀÌÅ¸¦ ÇØ¼®ÇÒ ¼ö ÀÖ´Â °°µµÀÇ ÆÐÅ°Áö´Ù. ±×´Â ÀÌ µÎ°¡Áö¸¦ ÇÕÄ¡°í mSQLÀ» Áö¿øÇÏµµ·Ï ÇÏ¿© PHP/FI¸¦ Å°»ý½ÃÄ×´Ù. PHP/FI´Â °ü¸¥ ¹ßÀüÇÏ¿´°í, ¸¹Àº »ç¶÷µéÀÌ À¯¿¡ °øÇåÇÏ¿´´Ù.

Á¤È®ÇÑ Åë°è´Â ¾øÁö¸¸ 1996³â ÈÄ¹Ý PHP/FI´Â Àü¼¼°èÀûÀ¸·Î ÃÖ¼ÒÇÑ 15,000°³ ÀÌ»óÀÇ À¥»çÀÌÆ®¿¡¼ »ç¿ëµÇ´Â °ÍÀ¸·Î ÃßÁ¤µÇ¾ú°í, 1997³â ÁßÀÝ¿¡ ±×¼ö´Â 50,000À¸·Î ´Ã¾î³µ´Ù. 1997³â ÁßÀ¹Ý PHP´Â ¶Ç´Ù¸¥ ÁßÀçÑ °¯È-¸¦ °¡Á®¿Ô´Ù. ÀÌ¶§°ÎÍ PHP´Â RasmusÀÇ °³ÀÎÀÌ ¾Æ´Ñ ÆÀ¿¡ ÀÇÇØ °³¹ßµÇ°í ÀÖ´Ù. »õ ÆÄ¼´Â Zeev Suraski¿Í Andi Gutmans°¡ Àç ÀÛ¼º¾µÇ¾ú°í, PHP Version 3¶ó´Â ÀÌ¸§À¸·Î ³ª¸²µ´Ù. ¸¹Àº PHP/FIÀÇ ±â´ÉµéÀÌ ¿Å°Ü¿Ô°í, ±×¿Ü¿¡µµ ¸¹Àº ±â´ÉµéÀÌ »õ·Î ÀÛ¼º¾µÇ¾ú´Ù.

1998³â ÁßÁÝÎ ÇöÀç PHP/FI³ª PHP3´Â C2ÀÇ StrongHold web server³ª RedHat Linux°°Àº ¿©·¯ »ó÷ÀûÀÎ Á¦Ç°°ú ÇÔ²² Á¦°øµÇ°í ÀÖÀ¸ç, Àü¼¼°èÀûÀ¸·Î ÃÖ¼ÒÇÑ 150,000°³ÀÇ À¥»çÀÌÆ®°¡ »ç¿ëÇÏ°í ÀÖ´Ù. ÀÌ ¼ö´Â ÀÎÅÍ³Ý¿¡¼ Netscape's flagship Enterprise serverÀÇ »ç¿ë¼ö¸´Ù ¸¹´Ù.

Chapter 2. PHP3ÀÇ ±â´É (PHP3 features)

HTTP ÀÎÁõ (HTTP authentication with PHP)

(¿ªÀÚÁÖ. HTTP authenticationÀÌ¶õ Web Client¿¡°Ô ID¿Í Password¸¦ ÀÛ·Â¹Þ¾Æ, ±× ID¿Í Password·Î Web¹®¼¿¡ Á¢±ÙÀ» Çã¹¡, °ÒÇãÇÏ´Â ±â´É» ¸»ÇÑ´Ù. ÀÚ¼¼È÷ ¾Ë°í ½Í´Ù¸é RFC1945ÀÇ Authentication °Î°ÐÀ» Âü°íÇÏ±â ¹Ù¶õ´Ù. http://pec.etri.re.kr/!qkim/HTTP/¿¡ ÇÑ±Û ¹ø¿ª ¹®¼µµ ÀÖÀ¸´Ï ÂüÁ¶ÇÏÀÚ.)

PHP¸¦ »ç¿ëÇÑ HTTP ÀÎÁõÀº Apache ¸ðµâ·Î »ç¿ëµÉ ¶§¸¸ µ¿ÀÛÇÑ´Ù. Apache ¸ðµâÀÇ PHP ½ºÅ©¸³Æ®¿¡¼- **Header()**¶ó´Â ÇÔ¼ö¸¦ »ç¿ëÇÏ¸é Client °ê¶ó¿ìÀú¿¡°Ô Username°ú Password¸¦ ÀÔ·Â¹Þ´Â À©µµ¿ì¸¦ ¶ç¿ìµµ·Ï ÇÏ´Â "ÀÎÁõ ¿ä±¸"("Authentication Required") ¸Þ½ÃÁö¸¦ º¸³»°Ô µÈ´Ù. ÀÏ´Ü »ç¿ëÀÚ°¡ Username°ú Password¸¦ ÀÔ·ÂÇÏ¸é, user name, password, authentication typeÀÇ °ªÀ» °¡Áö´Â $PHP_AUTH_USER, $PHP_AUTH_PW, $PHP_AUTH_TYPEÀÇ 3°³ÀÇ º¯¼ö¸¦ °¡Áö°í ÇØ´ç URL(PHP ½ºÅ©¸³Æ®¸¦ Æ÷ÇÔÇÑ)ÀÌ ´Ù½Ã ºÒ¸®¿öÁø´Ù. ÇöÀç authentication typeÀº "Basic"¸¸ÀÌ Áö¿øµÈ´Ù.

client authenticationÀ» »ç¿ëÇÏ¿© ±× ÀÔ·ÂµÈ °ªÀ» Ãâ·ÂÇÏ´Â ¿¹Á¦¸¦ ¾Æ·¡¿¡ ÀÖ´Ù. :
Example 2 - 1. HTTP Authentication example

```
<?php
  if(!$PHP_AUTH_USER) {
    Header("WWW-authenticate: basic realm=\"My Realm\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Text to send if user hits Cancel button\n"
    exit;
  } else {
    echo "Hello $PHP_AUTH_USER.<P>";
    echo "You entered $PHP_AUTH_PW as your password.<P>";
  }
?>
```

¿©·¯°ÐÀº $PHP_AUTH_USER ¿Í $PHP_AUTH_PW¸¦ ´Ü¼øÈ÷ Ãâ·ÂÇÏ´Â ´ë½Å, »ç¿ëÀÚÀÇ ½Ä°°À» À§ÇØ username°ú password¸¦ »ç¿ëÇÏ±â¸¦ ¿øÇÒ °ÍÀÌ´Ù. ÀÌ¶§´Â Database¿¡ QueryÇÏ°Å³ª Á÷Á¢ dbm ÆÄÀÏ¿¡¼- Ã£¾Æº¸¸é °¡´ÉÇÒ °ÍÀÌ´Ù.

Internet Explorer browser¿¡¼-´Â ¹ö±×·Î ÀÎÇØ HeaderÀÇ ¼ø¼-°¡ ¸Å¿ì ±î´Ù·Ó´Ù. *WWW-authenticate* header¸¦ HTTP/1.0 401 headerº¸´Ù ¸ÕÀú º¸³»´Â °ÍÀÌ ÇöÀç °¡´ÉÇÑ ¿ä·ÉÀ¸·Î º¸ÀÎ´Ù.

°ñ¹Ð¹øÈ£ÀÇ À¯ÃâÀ» ¿ì·ÁÇÏ¿© HTTP¿¡¼- Á¦°øÇÏ´Â ÀÎÁõ» »ç¿ëÇÏ´Â ´ë½Å ±âÁ¸ÀÇ ¿ÜºÎ ¸Ð«´ÏÁòÀ» »ç¿ëÇÒ °æ¿ì¿¡´Â ¹°·Ð PHP_AUTH º¯¼ö´Â ¼³Á¤µÇÁö ¾Ê´Â´Ù.

À¯ÀÇÇÒ Á¡ÀÌ ÀÎÁõÀÌ ÇÊ¿äÇÑ ÆäÀÌÁö¿Í ÀÎÁõÀÌ ÇÊ¿ä¾ø´Â ÆäÀÌÁö°¡ ¤°Àº ¼-¹ö»ó¿¡ Á¸ÀçÇÒ ¶§, ÀÎÁõÀÌ ÇÊ¿ä¾ø´Â URLÀ» controlÇÒ ¼ö ÀÖ´Â ´©±°¡°¡, ÀÎÁõÀÌ ÇÊ¿äÇÑ URLÀÇ ¾ÏÈ£¸¦ ÈÉÃÄº¸´Â °ÍÀº ¸·À» ¼ö ¾ø´Ù´Â Á¡ÀÌ´Ù.

Netscape¿Í Internet Explorer ¸ðµÎ 401ÀÀ´ä ÄÚµå¸¦ ¼-¹ö¿¡¼- ¹Þ°ÔµÇ¸é local browser windowÀÇ authentication cache¸¦ clearÇÑ´Ù. Áï, ÀÌ°ÍÀº "log out"°³³äÀÌ µÇ´Â °ÍÀ¸·Î, »ç¿ëÀÚµé·Î ÇÏ¿©±Ý username°ú ¾ÏÈ£¸¦ ´Ù½Ã ÀÔ·ÂÇÏ°Ô ÇÑ´Ù. ÀÏºÎ´Â ÀÌ°ÍÀ» loginÀÇ "time out"ÀÌ³ª "log-out"¹öÆ°À» ¸¸µå´Âµ¥ »ç¿ëÇÑ´Ù
Example 2 - 2. HTTP Authentication example forcing a new name/password

```
<?php
  function authenticate() {
    Header( "WWW-authenticate: basic realm='Test Authentication System'");
    Header( "HTTP/1.0 401 Unauthorized");
    echo "You must enter a valid login ID and password to access this resource\n";
    exit;
  }
  if(!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth, $PHP_AUTH_USER)) ) {
    authenticate();
  }
  else {
    echo "Welcome: $PHP_AUTH_USER<BR>";
    echo "Old: $OldAuth";
    echo "<FORM ACTION=\"$PHP_SELF\" METHOD=POST>\n";
    echo "<INPUT TYPE=HIDDEN NAME=\"SeenBefore\" VALUE=\"1\">\n";
    echo "<INPUT TYPE=HIDDEN NAME=\"OldAuth\" VALUE=\"$PHP_AUTH_USER\">\n";
    echo "<INPUT TYPE=Submit VALUE=\"Re Authenticate\">\n";
    echo "</FORM>\n";
}
?>
```

ÀÌ µ¿ÀÛº HTTP Basic authentication ±âÁØ¿¡ ¿ä±¸µÇ´Â °ÍÀº ¾Æ´Ï¹Ç·Î ¿©·¯°ÐÀº ÀÌ°Í¿¡ ÀÇÁ¸ÇÏ¿©¼-´Â ¾ÈµÈ´Ù. Lynx¿¡¼-´Â 401 ¼-¹ö ÀÀ´äÀ» ¹Þ¾Æµµ authentication Á¤º¸¸¦ clearÇÏÁö ¾Ê´Â´Ù. µû¶ó¼- BackÀ» ´©¸£°í ´Ù½Ã Forward ¹öÆ°À» ´-·¯¼- ´Ù½Ã Á¢±ÙÇÒ ¼ö ÀÖ´Ù.

¶ÇÇÑ ÀÌ Authentication ±â´ÉÀº Microsoft's IIS server¿¡¼- CGI versionÀÇ PHP¸¦ »ç¿ëÇÒ °æ¿ì¿¡´Â IISÀÇ Á¦¾àÀ¸·ÎÎÇØ µ¿ÀÛÇÏÁö ¾Ê´Â´Ù.

PHP¿¡¼- GIFÀÇ »ý¼º (GIF creation with PHP)

PHP´Â ´ÜÁö HTMLÃâ·ÂÀ» ¸¸µå´Â °Í¸¸ ÇÒ ¼ö ÀÖ´Â °ÍÀÌ ¾Æ´Ï´Ù. PHP´Â GIF image fileµµ ¸¸µé ¼ö ÀÖÀ» ÀÖÀ» »Ó¾Æ´Ï¶ó, »ç¿ëÇÏ±â¿¡ Æí¸®ÇÑ GIF image stream±îÁö ¸¸µé ¼ö ÀÖ´Ù. ÀÌ¸¦ À§ÇØ¼- ¿©·¯°ÐÀº PHP¸¦ ÄÄÆÄÀÏ ÇÒ ¶§, image ÇÔ¼ö¸¦ °¡´ÉÇÏ°í ÀÖ´Â GD ¶óÀÌºê·¯¸®¸¦ Æ÷ÇÔÇÏ¿©¾ß ÇÑ´Ù.
Example 2 - 2. GIF creation with PHP

```
<?php
    Header("Content-type: image/gif");
```

```
    $string=implode($argv," ");
    $im = imagecreatefromgif("images/button1.gif");
    $orange = ImageColorAllocate($im, 220, 210, 60);
    px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im,3,$px,9,$string,$orange);
    ImageGif($im);
    ImageDestroy($im);
?>
```

À§ÀÇ ¿¹Á¦´Â <img src="button.php3?text">¿Í °°Àº tagÀÌ ÀÖ´Â ÆäÀÌÁö·Î°ÎÅÍ ºÒ·ÁÁö°Ô µÉ °ÍÀÌ´Ù. ±×·¯¸é À§¿¡ÀÖ´Â button.php ½ºÅ©¸³Æ®´Â "text"¶ó´Â ¹®ÀÚ¿À» "images/button1.gif"¿¡ ¿À¹ö·¹ÀÌ ½ÃÄÑ °á°ú image¿¡ Ââ·ÂÇÑ´Ù. ÀÌ·¸°Ô ÇÏ¸é ¹öÆ°¿¡ µé¾î°¡´Â ±Û¾¾¸¦ ¸Å¹ø ¼Õ½±°Ô ¹Ù²ã ¾µ ¼ö ÀÖ°í, ¶ÇÇÑ ¸Å¹ø ÀÌ¹ÌÁö ÆÄÀÏÀ» ¸¸µé ÇÊ¿ä°¡ ¾ø¾î È¿À²ÀûÀÌ°í °£´ÜÇÏ´Ù.

File upload support

PHP´Â RFC-1867À» Áö¿øÇÏ´Â °ê¶ó¿ìÀú·Î°ÎÅÍ ÆÄÀÏÀ» ¾÷·Îµå ¹ÞÀ» ¼ö ÀÖ´Â ±â´ÉÀÌ ÀÖ´Ù. ÀÌ ±â´ÉÀ» »ç¿ëÇÏ¸é Text°Ð ¾Æ´Ï¶ó BinaryÆÄÀÏµµ ¾÷·Îµå°¡ °¡´ÉÇÏ´Ù. ¿©·¯°ÐÀº PHP's authetication°ú ³í¸®ÀûÀÎ ÇÔ¼öµéÀ» »ç¿ëÇÏ¿©, Upload°¡ °¡´ÉÇÑ »ç¿ëÀÚ¿Í, ÆÄÀÏÀÌ UploadµÈ ÈÄ¿¡ ÇØ¾ß ÇÒ ÀÏÀ» ¹Ýµå½Ã Á¤ÇØ µÎ¾î¾ß ÇÑ´Ù. (¿ªÀÚÁÖ. RFC-1867Àº Netscape 3.0ÀÌ»ó, Explorer 4.0ÀÌ»ó¿¡¼ Áö¿øÇÑ´Ù. Netscape 2.x´ëÀÇ ÀÏ°Î »óÀ§¹öÀ¯µµ Áö¿øÇÏ°í, Explorer 3.02´Â Patch°¡ ³ª¿Í ÀÖÀ¸¹Ç·Î patch½ÃÅ°¸é °¡´ÉÇÏ´Ù.)

ÆÄÀÏ ¾÷·Îµå È¸éÀº ´ÙÀ½°ú °°Àº Á» Æ¯°°ÇÑ ÆûÀ» ¸¸µé¾î ¶ç¿ï ¼ö ÀÖ´Ù. :

Example 2-3. File Upload Form

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

¿©±â¼_URL_Àº PHP htmlÆÄÀÏÀÌ¾î¾ß ÇÕ´Ï´Ù. hidden ÇÊµåÀÎ MAX_FILE_SIZE´Â File input ÇÊµåµé°¸´Ù ¼±ÇàµÇ¾î¾ß ÇÕ´Ï´Ù. ÀÌ °ªÀº PHP htmlÀÌ ¹Þ¾ÆµéÀÌ´Â ÃÖ´ë ÆÄÀÏ Å©±â¸¦ Byte´ÜÀ§·Î ³ªÅ¸³À´Ï´Ù. ¾÷·Îµå°¡ ¼º°øÀûÀ¸·Î µÇ¸é ÁöÁ¤µÈ ÆÄÀÏ¿¡´Â ´ÙÀ½°ú °°Àº º¯¼öµéÀÌ Á¤ÀÇµÇ¾î Áø´Ù. :

$userfile - ¾÷·ÎµåµÈ ÆÄÀÏ ³»¿ëÀÌ ÀúÀåµÇ¾î ÀÖ´Â ¼¹öÀÇ ÀÓ½Ã ÆÄÀÏ¸í

$userfile_name - ¾÷·ÎµåÇÑ ½Ã½ºÅÛ¿¡¼ »ç¿ëÇÏ´Â ÆÄÀÏÀÇ ¿ø·¡ ÀÌ¸§.

$userfile_size - byte´ÜÀ§ÀÇ UploadµÈ ÆÄÀÏÀÇ Å©±â.

$userfile_type - ¸¸¾à browser°¡ ¾÷·ÎµåµÈ ÆÄÀÏÀÇ mime Çü½ÄÀ» ¾È´Ù¸é, ±× mime Çü½Ä. (Ex. "image/gif").

À§ º¯¼öÀÇ "$userfile"°Î°ÐÀº upload form¿¡¼ TYPE=filedÀ» °¡Áø INPUT ÇÊµåÀÇ ÀÌ¸§ÀÌ µÈ´Ù. À§ÀÇ ¿¹Á¦¿¡¼ ¿ì¸®´Â ±× ÀÌ¸§À° "userfile"ÀÌ¶ó°í Á¤Çß´Ù.

FileÀº ±â°»ÀûÀ¸·Î ¿ì¼± ¼¹öÀÇ default temporary directory¿¡ ÀúÀåµÈ´Ù. ÀÌ µðÆúÆ® µð·ºÅä¸®´Â PHP°¡ µ¹¾Æ°¡´Â ÄÄÇ»ÅÍ ÀÇ È¯°æ°¯¼ö "TMPDIR"À» ¼³Á¤ÇÏ¿© °¯°æÇÒ ¼ö ÀÖ´Ù. ÀÌ¸¦ PHP ½ºÅ©¸³Æ® ¾È¿¡¼ PutEnv() ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© °¯°æÇÏ´Â °ÍÀº µ¿ÀÛÇÏÁö ¾Ê´Â´Ù.

¾÷·ÎµåµÈ ÆÄÀÏÀ» ´Ù·ç´Â PHP ½ºÅ©¸³Æ®´Â ÇØ´ç ÆÄÀÏÀ» °¡Áö°í ÇÏ´Â ÀÛ¾÷À» ¼³Á¤ÇØ ÁÙ ÇÊ¿ä°¡ ÀÖ´Ù. ¿¹¸¦µé¾î, ¿©·¯°ÐÀº $file_size°¯¼ö¸¦ »ç¿ëÇÏ¿© ³Ê¹« Å©°Å³ª ³Ê¹« ÀÛÀº ÆÄÀÏÀ» ¹ö¸± ¼öµµ ÀÖ´Ù. ¶ÇÇÑ ¿©·¯°ÐÀº $file_type°¯¼ö¸¦ °¡Áö°í Æ¯Á¤ÇÑ Å¸ÀÔ¿¡ ¸ÂÁö ¾Ê´Â ÆÄÀÏÀ» ¹ö¸± ¼ö ÀÖ´Ù. ¾î¶² ·ÎÁ÷ÀÌ´ø, ¿©·¯°ÐÀº ÀÓ½Ã µð·ºÅä¸®¿¡ ÀÖ´Â ÆÄÀÏÀ» Áö¿ì°Å³ª ³ªÁß¿¡ ÇÊ¿äÇÒ °æ¿ì¿¡´Â ´Ù¸¥ °÷¿¡ ÀÌµ¿½ÃÄÑ¾ß ÇÑ´Ù.

CERN httpd ¼¹ö´Â client·Î°ÎÅÍ ÀÔ·Â¹ÞÀº mime headerÀÇ ¾ÕÂÊ ¿©¹éÀ» ¸ðµÎ strip off ½ÃÄÑ¹ö¸®¹Ç·Î, CERN httpd ¼¹ö ¿¡¼´Â File Upload ±â´ÉÀÌ µ¿ÀÛÇÏÁö ¾Ê´Â´Ù.

PUT Method Support

PHP´Â Netscape Composer³ª W3C Amaya°°Àº Å¬¶óÀÌ¾ðÆ®¿¡ ´ëÇØ HTTP PUT ¹æ½Ä(method)À» Áö¿øÇÑ´Ù. PUT ¿äÃ» (request)´Â file upload°¸´Ù ÈÎ¾À ½±´Ù. ´ÜÁö ´ÙÀ½°ú °°ÀÌ ÇÏ¸é µÈ´Ù. :

PUT /path/filename.html HTTP/1.1

ÀÌ°ÍÀº °¸Åë ¿ø°Ý Å¬¶óÀÌ¾ðÆ®°¡ °¸³½ ³»¿ëÀ» À¥ Æ®¸®¸® ¹ØÀÇ /path/filename.html·Î ÀúÀåÇÏ¶ó´Â ÀÇ¹ÌÀÌ´Ù. ±×·±µ¥ ¿©·¯°ÐÀÇ À¥ Æ®¸®¸®¹Ø¿¡ ÀÖ´Â ÆÄÀÏµéÀ» ¾Æ¹«³ª µ¤¾î ¾µ ¼ö ÀÖ´Ù´Â °ÍÀº Apache³ª PHP¿¡ ÀÖ¾î¼ È®½ÇÈ÷ ÁÁÁö ¾ÊÀº »ý°¢ÀÌ´Ù. µû¶ó¼ ÀÌ¿Í °°Àº ¿äÃ»¸¦ ´Ù·ç±â Àü¿¡, ¿ì¼± À¥ ¼¹ö¿¡°Ô ÀÌ·± ¿äÃ»¸¦ ´Ù·ç´Â PHP ½ºÅ©¸³Æ®¸¦ ¹Ì¸® ÁöÁ¤ÇØ ÁÖ¾î¾ß ÇÑ´Ù. ¾ÆÆÄÄ¡¿¡¼´Â *Script* Áö½ÃÀÚ·Î ±× »ç¿ëÀ» ÁöÁ¤ÇÑ´Ù. ÀÌ Áö½ÃÀÚ´Â Apache ¼³Á¤ ÆÄÀÏÀßÀÇ ¾î´À À§Ä¡¿¡ ÀÖ¾îµµ ±¦ÂúÀ¸¸ç, °¸Åë <Directory> °í·Ï ¾ÈÀÌ³ª <Virtualhost> °í·Ï ¾È¿¡ À§Ä¡ÇÏ´Â °ÍÀÌ ÀÏ¹ÝÀûÀÌ´Ù. °¸Åë ´ÙÀ½°ú °°ÀÌ ¾²¿©Áø´Ù. :

Script PUT /put.php3

ÀÌ°ÍÀº ÀÌ ¶óÀÎÀ» À§Ä¡½ÃÅ² °í·ÏÀÇ URI¿¡ ÇØ´çµÇ´Â ¸ðµç PUT ¿ä±¸¸¦ put.php3 ½ºÅ©¸³Æ®¿¡°Ô Àü´ÞÇÏ¶ó°í Apache¿¡°Ô ¾Ë·Á ÁØ´Ù. ¹°·Ð ÀÌ °æ¿ì .php3 È®ÀåÀÚ¿¡ ´ëÇÏ¿© PHP¼³Á¤ÀÌ ¿Ï·áµÇ¾î ÀÖ°í, PHP°¡ ÀÛµ¿ÁßÀÌ¾î¾ß ÇÑ´Ù.

put.php3 ÆÄÀÏ³»¿¡¼´Â º¸Åë ´ÙÀ½°ú °°ÀÌ »ç¿ëÇÒ ¼ö ÀÖ´Ù. :

<? copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>

À§ÀÇ ¸í·ÉÀº ÇØ´ç ÆÄÀÏÀ» ¿ø°Ý Å¬¶óÀÌ¾ðÆ®°¡ ¿äÃ»ÇÑ À§Ä¡¿¡ º¹»çÇÏ´Â °ÍÀÌ´Ù. ¾Æ¸¶ ¿©·¯ºÐÀº º¹»çÇÏ±â Àü¿¡ »ç¿ëÀÚ¸¦ È®ÀÎÇÏ°Å³ª ÆÄÀÏÀ» °Ë»çÇÏ´Â µîÀÇ ±â´ÉÀ» ¿øÇÒ °ÍÀÌ´Ù. ¿©±â¼ ¾Ë ¼ö ÀÖ´Â °ÍÀº PHP°¡ PUT-methodÀÇ ¿ä±¸¸¦ ¹Þ¾ÒÀ» ¶§ File Upload ±â´É°ú ¸¶Âù°¡Áö·Î ÀÓ½Ã ÆÄÀÏ¿¡ ÇØ´ç ³»¿ëÀ» ÀúÀåÇÑ´Ù´Â °ÍÀÌ´Ù. Áï, ¿ä±¸°¡ ³¡³ª°Ô µÇ¸é ÀÓ½ÃÆÄÀÏÀº Áö¿öÁø´Ù. µû¶ó¼ PUTÀ» ´Ù·ç´Â PHP ½ºÅ©¸³Æ®´Â ÇØ´ç ÆÄÀÏÀ» ¾îµð ´Ù¸¥ °÷¿¡ º¹»çÇØ µÎ¾î¾ß ÇÑ´Ù. ÀÓ½ÃÆÄÀÏÀÇ ÀÌ¸§Àº $PHP_PUT_FILENAME ÀÌ¶ó´Â º¯¼ö¿¡ ÀúÀåµÇ°í, $REQUEST_URI º¯¼ö¿¡ Å¬¶óÀÌ¾ðÆ®¿¡¼ º¸³»¿Â ÀúÀåÇÒ ÆÄÀÏÀÇ °æ·Î¿Í ÀÌ¸§ÀÌ ÀúÀåµÈ´Ù.(Apache°¡ ¾Æ´Ñ À¥¼¹ö¿¡¼´Â ¸ð¾çÀÌ Á¶±Ý ´Þ¶óÁø´Ù.) ¹°·Ð ¿©·¯ºÐÀº ÀÌ °æ·Î¸¦ ¹«½ÃÇÏ°í ÀÌ°ÍÀ» ¾Æ´Ñ ÀüÇô ´Ù¸¥ À§Ä¡¿¡ ´Ù¸¥ ÆÄÀÏ¸íÀ» »ç¿ëÇÒ ¼öµµ ÀÖ´Ù.

HTTP cookie support

PHP´Â HTTP Cookie¸¦ Åõ¸íÇÏ°Ô(transparently) Áö¿øÇÑ´Ù. Cookie ¸ÞÄ«´ÏÁòÀº Æ®·¡Å·ÀÌ³ª »ç¿ëÀÚ ½Äº° µîÀ» À§ÇØ ¿ø°Ý browser¿¡ ÀúÀåµÈ µ¥ÀÌÅÍ¸¦ µ¹·Á ¹Þ´Â °úÁ¤À» ¸»ÇÑ´Ù. ¿©·¯ºÐÀº cookie¸¦ ¼³Á¤ÇÏ±â À§ÇØ **setcookie()** ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼ö ÀÖ´Ù. Cookie´Â HTTP Çì´õÀÇ ÇÑ ºÎºÐÀÌ¹Ç·Î, SetCookie() ÇÔ¼ö´Â ºê¶ó¿ìÀú·Î º¸³»´Â ¾î¶² µ¥ÀÌÅÍº¸´Ùµµ ¾Õ¿¡ »ç¿ëÇØ¾ß ÇÑ´Ù. ÀÌ Á¦¾àÀº **Header()** ÇÔ¼ö¿Í °°Àº Á¦¾àÀ¸·Î º¸¸é µÈ´Ù.

´ç½ÅÀÌ º¸³»ÁØ ¸ðµç cookie´Â ÀÚµ¿ÀûÀ¸·Î GETÀÌ³ª POST ¹æ½Ä µ¥ÀÌÅÍ¿Í °°Àº PHP º¯¼ö·Î º¯È¯µÈ´Ù. ¸¸¾à ¿©·¯ºÐÀÌ µ¿ÀÏÇÑ cookie¿¡ ¿©·¯ °ªÀ» ÀúÀåÇÏ°í ½Í´Ù¸é CookieÀÌ¸§¿¡ *[]*¸¦ ´õÇÏ¸é µÈ´Ù. ÀÚ¼¼ÇÑ °ÍÀº **setcookie()** ÇÔ¼ö ¼³¸íÀ» ÂüÁ¶ÇÏ ÀÚ.

Database support

PHP´Â native mode¿Í ODBC¸¦ ÅëÇØ ¼ö¸¹Àº database¸¦ »ç¿ëÇÒ ¼ö ÀÖ´Ù.

Adabas D	MySQL
dBase	Oracle
Empress	PostgreSQL
FilePro	Solid
Informix	Sybase
InterBase	Velocis
mSQL	Unix dbm

Regular expressions

Regular expressionÀº PHP¿¡¼ º¹ÀâÇÑ ¹®ÀÚ¿ Á¶ÀÛÀ» À§ÇØ »ç¿ëÇÑ´Ù. regular expressionÀ» Áö¿øÇÏ±â À§ÇØ ´ÙÀ½ÀÇ ÇÔ¼ö°¡ »ç¿ëµÈ´Ù. :

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

À§ÀÇ ÇÔ¼öµéÀº regular expression ¹®ÀÚ¿À» Ã¹ ÀÎ¼ö(argument)·Î °¡Áø´Ù. PHP´Â Posix 1003.2¿¡¼ Á¤ÀÇµÈ Posix È®Àå regular expressionÀ» »ç¿ëÇÑ´Ù. Posix regular expressions¿¡ ´ëÇÑ ÀÚ¼¼ÇÑ ¼³¸íÀº PHP distributionÀÇ regexµð·ºÅä¸®¿¡ ÀÖ´Â regex man page¸¦ ÂüÁ¶ÇÏ±â ¹Ù¶õ´Ù.

Example 2-4. Regular expression examples

```
ereg("abc",$string);
/* Returns true if "abc"
   is found anywhere in $string. */

ereg("^abc",$string);
/* Returns true if "abc"
   is found at the beginning of $string. */

ereg("abc$",$string);
/* Returns true if "abc"
   is found at the end of $string. */

eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
/* Returns true if client browser
```

```
   is Netscape 2, 3 or MSIE 3. */

ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)",
     $string,$regs);
/* Places three space separated words
   into $regs[1], $regs[2] and $regs[3]. */

ereg_replace("^","<BR>",$string);
/* Put a <BR> tag at the beginning of $string. */

ereg_replace("$","<BR>",$string);
/* Put a <BR> tag at the end of $string. */

ereg_replace("\n","",$string);
/* Get rid of any carriage return
   characters in $string. */
```

Error handling

PHP¿¡´Â ´ÙÀ½°ú °°Àº 4°¡ÁöÀÇ ¿¡·¯¿Í °æ°í ÇüÅÂ°¡ ÀÖ´Ù.

1 - Normal Function Errors (ÀÏ¹ÝÀûÀÎ ¿¡·¯)

2 - Normal Warnings (ÀÏ¹ÝÀûÀÎ °æ°í)

4 - Parser Errors (¹®¹ý ¿¡·¯)

8 - Notices (¹«½ÃÇÒ ¼ö ÀÖ´Â °æ°íÀÌ³ª ¹ö±×ÀÇ °¡´É¼ºÀ» °¡Áö°í ÀÖ´Â °æ¿ì)

error reporting levelÀº À§ÀÇ 4°³ÀÇ ¼ýÀÚ°¡ ´õÇØÁ®¼- °áÁ¤µÈ´Ù. ±â°» error reporting level Àº 7·Î 1+2+4°¡ µÇ°í, ÀÌ´Â Notice¸¦ Á¦¿ÜÇÑ ¸ðµç °ÍÀ» ·¹Æ÷Æ®ÇÏ¶ó´Â ÀÇ¹ÌÀÌ´Ù. ÀÌ ·¹°§Àº php3.ini ÆÄÀÏ¿¡¼- error_reporting Áö½ÃÀÚ·Î °áÁ¤ÇÒ ¼ö ÀÖ´Ù. È¤Àº, ¾ÆÆÄÄ¡ÀÇ httpd.conf ÆÄÀÏ¿¡¼- php3_error_reporting Áö½ÃÀÚ·Î °áÁ¤ÇÏ°Å³ª, ½ºÅ©¸³Æ® ½ÇÇà ÁßÂ¿¡ **error_reporting()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ¹Ù²ãÁÙ ¼öµµ ÀÖ´Ù.

¸ðµç PHP Ç¥Çö½Ä(expression)Àº "@"¸¦ ¾Õ¿¡ °ÙÀÌ°í È£ÃâµÇ¸é error reportingÀ» ÇÏÁö ¾Ê´Â´Ù. ¸¸¾à track_errors ±â´ÉÀÌ Enabled·Î µÇ¾î ÀÖ´Âµ¥ ÇØ´ç ¹®Àå¿¡¼- ¿¡·¯°¡ ¹ß»ýÇß´Ù¸é, $php_errormsg¶ó´Â Àü¿ª°¯¼ö¿¡¼- ¿¡·¯ ¸þ½ÃÁö¸¦ Ã£À» ¼ö ÀÖ°Ô À» °ÍÀÌ´Ù.

Connection Handling

ÀÌ ±â´ÉÀº PHP version 3.0.7 ÀÌÈÄ°ÎÅÍ Àû¿ëµÈ °ÍµéÀÌ´Ù.

³»°ÌÀûÀ¸·Î PHP´Â ¿¬°áÀÇ »óÅÂ¸¦ ´ÙÀ½ 3°¡ÁöÁß ÇÏ³ª·Î ¼³Á¤ÇÑ´Ù. :

0 - NORMAL

1 - ABORTED

2 - TIMEOUT

PHP ½ºÅ©¸³Æ®°¡ ÀÏ¹ÝÀûÀ¸·Î ½ÇÇàµÇ°í ÀÖÀ» ¶§´Â NORMAL »óÅÂ°¡ activeµÈ´Ù. ¸¸¾à ¿ø°Ý Å¬¶óÀÌ¾ðÆ®°¡ ¿¬°áÀ» ²÷°ÔµÇ¸é ABORTED »óÅÂ flag°¡ ÄÑÁö°Ô µÈ´Ù. °¸Åë Å¬¶óÀÌ¾ðÆ®°¡ ¿¬°áÀ» ²÷´Â °ÍÀº »ç¿ëÀÚ°¡ STOP ¹öÆ°À» ´·¯¶À» ¶§ ÀÏ¼î³-´Ù. ¸¸¾à PHP°¡ ¼³Á¤ÇÑ ½Ã°£ Á¦ÇÑ¿¡ (**set_time_limit()**À» °¸®®°Ô µÇ¸é TIMEOUT »óÅÂ flag°¡ ÄÑÁö°Ô µÈ´Ù.

¿©·¯°ÐÀº Å¬¶óÀÌ¾ðÆ®°¡ ¿¬°áÀ» ²÷¾úÀ» ¶§ ½ºÅ©¸³Æ®°¡ ¼öÇàÀ» ÁßÄ´ÜÇÒ °ÍÀÎ°¡ ¾Æ´Ñ°¡¸¦ °áÁ¤ÇÏ¿©¾ß ÇÑ´Ù. ¶§¶§·Î ¿ø°Ý °ê¶ó¿ìÀú°¡ ¾î¶² °á°úµµ ¹Þ¾Æµé¾ÌÁö ¾Ê´õ¶óµµ ½ºÅ©¸³Æ®¸¦ ³¡±îÁö ½ÇÇàÇÏ´Â °ÍÀÌ Æí¸®ÇÔ ¶§°¡ ¸¹´Ù. ±â°» µ¿ÀÛÀº Å¬¶óÀÌ¾ðÆ®°¡ ¿¬°áÀ» ²÷À¸¸é ½ºÅ©¸³Æ®µµ ÁßÄ´ÜÇ´Â °ÍÀÌ´Ù. ÀÌ µ¿ÀÛÀº ignore_user_abort¶ó´Â php3.ini Áö½ÃÀÚ(directive)³ª, php3_ignore_user_abort¶ó´Â Apache .confÀÇ Áö½ÃÀÚ·Î ¼³Á¤ÀÌ °¡´ÉÇÏ°í, **ignore_user_abort()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ¼³Á¤ÇÒ ¼öµµ ÀÖ´Ù. ¸¸¾à ¿©·¯°ÐÀÌ PHP¿¡°Ô »ç¿ëÀÚ°¡ ³»¸¸ ÁßÁö ¸í·ÉÀ» ¹«½ÃÇÏ¶ó°í »ç¿ëÀÚ°¡ ¸»Çß´Ù¸é, »ç¿ëÀÚÀÇ ÁßÁö ½ÅÈ£´Â ¿©·¯°ÐÀÇ ½ºÅ©¸³Æ®¸¦ Á¾·á½ÃÅ²´Ù. ´Ü, **register_shutdown_function()**À» »ç¿ëÇÏ¿© shutdown ÇÔ¼ö¸¦ µî·ÏÇÇ³õÀº °æ¿ì´Â ¿¹¿Ü°¡ µÈ´Ù. shutdown ÇÔ¼ö¸¦ »ç¿ëÇÏ¸é, »ç¿ëÀÚ°¡ STOP¹öÆ°À» ´·¯¶À» ¶§, ¿©·¯°ÐÀÇ ½ºÅ©¸³Æ®´Â ³²Àº Ãâ·ÂÀ» ÇÏ·Á°í ÇÏ°í, PHP´Â ¿¬°áÀÌ ÁßÄ´ÜµÇ¾úÀ½» ÀÎÁöÇÏ¿© shutdown ÇÔ¼ö¸¦ È£ÃâÇÏ°Ô µÈ´Ù. shutdown ÇÔ¼ö´Â ¿©·¯°ÐÀÇ ½ºÅ©¸³Æ®°¡ Á»óÀû À¸·Î Á¾·áµÇ¾úÀ» ¶§µµ È£Ãâ µÈ´Ù. µû¶óº- Å¬¶óÀ¾ðÆ®ÀÇ ¿¬°á ÁßÄ¶§¿Í ÀÏ¹ÝÀûÀÎ Á¾·á½Ã¿¡ ´Ù¸¥ µ¿ÀÛ» ¿øÇÑ´Ù¸é **connection_aborted()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¸é µÈ´Ù. ÀÌ ÇÔ¼ö´Â ¿¬°áÀÌ ÁßµÇ¾ú´Ù¸é true¸¦ ¹ÝÈ¯ÇÑ´Ù.

¿©·¯°ÐÀÇ ½ºÅ©¸³Æ®´Â ³»ºÎÈ Å¸ÀÌ¸Ó¿¡ ÀÇÇØ Á¾·áµÉ ¼ö ÀÖ´Ù. ±â°» timeout ½Ã°£Àº 30ÃÊ·Î µÇ¾î ÀÖ´Ù. ÀÌ°ÍÀº max_execution_timeÀÌ¶ó´Â php3.ini Áö½ÃÀÚ(directive)³ª µ¿ÀÏÇÑ ³»¿ëÀÇ php3_max_execution_timeÀÌ¶ó´Â Apache .conf Áö½ÃÀÚ¿¡ ÀÇÇØ ´Ù¸£°Ô ¾³Á¤ÀÌ °¡´ÉÇÏ°í **set_time_limit()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ¼³Á¤ÇÒ ¼öµµ ÀÖ´Ù. ½Ã°£ÀÌ ´Ù µÇ¸é ½ºÅ©¸³Æ®

´Â ÁßˊÜµÇ°í, ÀÌ ½ºÅ©´Â À§¿¼- ¼³¸íÇÑ Å¬¶óÀÌ¾ðÆ®¿ÍÀÇ ¿¬°áÀÌ Á¾·áµÈ °ÍÃ³·³ µ¿ÀÛÇÑ´Ù. µû¶ó¼- shutdown ÇÔ¼ö°¡ µî·ÏµÇ ¾î ÀÖ´Ù¸é ÀÌ ÇÔ¼ö°¡ ½ÇÇàµÈ´Ù. shutdown ÇÔ¼ö³»¿¡¼- ÀÌ ÇÔ¼ö°¡ timeout¿¡ ÀÇÇØ È£ÃâµÈ °ÍÀÎ°¡¸¦ Æ˜ÜÇÏ·Á¸é **connection_timeout()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¸é µÈ´Ù. ÀÌ ÇÔ¼ö´Â timeout¿¡ ÀÇÇØ shutdown ÇÔ¼ö°¡ È£ÃâµÇ¾ú´Ù¸é true¸¦ ¹ÝÈ¯ÇÑ´Ù.

ÇÑ°¡Áö À¯ÀÇÇÒ »çÇ×Àº ABORTED¿Í TIMEOUT »óÅÂ´Â µÎ °³°¡ µ¿½Ã¿¡ activeµÉ ¼ö ÀÖ´Ù´Â °ÍÀÌ´Ù. ÀÌ°ÍÀº PHP°¡ »ç¿ëÀÚ ÁßˊÜÀ» ¹«½ÃÇÏµµ·Ï ¼³Á¤ÇØ ³õ¾ÒÀ» ¶§ °¡´ÉÇÏ´Ù. ¶Ç ´Ù¸¥ °æ¿ì´Â »ç¿ëÀÚ°¡ ¿¬°áÀ» ²÷Áö´Â ¾Ê¾ÒÁö¸¸ ¿¬°á¿¡ ¾î¶² ¹®Á¦°¡ »ý°Ü ¿¬°áÌ ¸Á°¡Áø °æ¿ì·Î, ÀÌ ¶§µµ ½ºÅ©¸Æ®´Â °è¼Ó ½ÇÇàµÈ´Ù. ÀÌ·± °æ¿ì¿¡ ½ºÅ©¸Æ®´Â °è¼Ó½ÇÇàµÇ°í, ½Ã°£ Á¦ÇÑ¿¡ °É·Á ½ºÅ©¸Æ®ÀÇ ¼öÇàÀÌ ÁßˊÜµÇ°í shutdown ÇÔ¼ö°¡ È£ÃâµÇ¸é, **connection_timeout()**°ú **connection_aborted()** ÇÔ¼ö ¸ðµÎ true¸¦ ¹ÝÈ¯ÇÏ°Ô µÉ´Ù. ¿©·¯°ÐÀº **connection_status()** ¶ó´Â ÇÔ¼ö¸¦ »ç¿ëÏ¿© µÎ°¡Áö »óÅÂ ¸ðµÎ¸¦ °Ë»çÇÒ ¼ö ÀÖ´Ù. ÀÌ ÇÔ¼ö´Â °¢ »óÅÂ¿¡ ´ëÇÑ Á¤º¸¸¦ bit´ÜÀ§·Î °¡Áö°í ÀÖ´Ù. ¿¹¸¦µé¾î À§¿°°ÀÌ ABORT¿Í TIMEOUT µÎ °³ÀÇ »óÅÂ°¡ ¸ðµÎ activeµÇ¾î ÀÖ´Ù¸é ÀÌ ÇÔ¼ö´Â 3Àº ¹ÝÈ¯ÇÑ´Ù.

PHP source viewer

Chapter 3. ¼³Ä¡ (Installation)

ÀÌ ÀåÀº PHP3¸¦ ¼³Ä¡ÇÏ°í ¼³Á¤ÇÏ´Â ¹æ¹ý¿¡ ´ëÇØ ¼³¸íÇÏ°í ÀÖ´Ù. ÇÊ¿äÇÑ Áö½Ä°ú ¼ÒÇÁ¦®¿þ¾î :

±â°»ÀûÀÎ UNIX »ç¿ë¹ý ("make"¿Í C ÄÄÆÄÀÏ·¯ÀÇ »ç¿ë)

ANSI C ÄÄÆÄÀÏ·¯

web ¼-¹ö (¹Ýµå½Ã ¾Ë¾Æ¾ß ÇÔ)

Unix¿¡¼- ¼Ò½º·Î ¼³Ä¡ (Installing From Source on UNIX)

Downloading Source

¿©·¯°ÐÀº ÃÖ½Å¹öÀüÀÇ ¼Ò½º¸¦ http://www.php.net¿¡¼- ³»·Á ¹ÞÀ» ¼ö ÀÖ´Ù.

°ü¸¥ ¼³Ä¡ ¹æ¹ý (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-3.0.x.tar.gz
4.  tar xvf php-3.0.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-3.0.x
8.  ./configure --with-mysql=/usr/local/mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --prefix=/www --activate-module=src/modules/php3/libphp3.a
13. make
14. make install
   ÀÌ ´Ü°è ´ë½Å httpd ÀÌÁø ÆÄÀÏÀ» ±âÁ¸ ÀÇ ÆÄÀÏ¿¡ µ¤¾î ¾µµ ¾öµµ ÀÖ´Ù.
   ÀÌ ÀÛ¾÷À» ÇÏ·Á¸é ¿ì¼± httpd ¼-¹ö¸¦ ²ô´Â °ÍÀ» ÀØÁö ¸»·µ·Ï ÇïÀÚ.
15. cd ../php-3.0.x
16. cp php3.ini-dist /usr/local/lib/php3.ini
   ¿©·¯°ÐÀº ÀÌÁ¦ PHPÀÇ ¼³Á¤À» °íÄ¡±âÀ§ÇØ /usr/local/lib/php3.ini¸¦ ¼öÁ¤ÇÏ¸é µÈ´Ù.
  ¸¸¾à ÀÌ ÆÄÀÏÀÇ À§Ä¡¦ ¿Å±â°í ½Í´Ù¸é À§ÀÇ 8´Ü°è¿¡¼- --with-config-file=/path ¿É¼ÇÀ» ÁØ´Ù.
17. httpd.conf ÆÄÀÏÀÌ³ª srm.conf ÆÄÀÏ¿¡ ´ÙÀ½À» Ã·°¡ÇÑ´Ù.
            AddType application/x-httpd-php3 .php3
  ¿©±â¼- ¿©·¯°ÐÀº .php3¿Ü¿¡ ´Ù¸¥ È®ÀåÀÚ¦ »ç¿ëÇÒ ¼öµµ ÀÖ´Ù.
  ÇïÁö¸¸ ¸¸¾À» ç¶÷µéÀÌ »ç¿ëÇÏ´Â ë·Î ±×É .php3¸¦ »ç¿ëÇÒ °ÍÀ» ±ÇÇÑ´Ù.
18. ÀÏ¹ÝÁÚÀÎ ¹æ½Ä´ë·Î Apache ¼-¹ö¸¦ ½ÃÀÛÇÑ´Ù.
   (¹Ýµå½Ã stop°ú startÇÏ¿© Àç½ÃµµÇÏ¿©¾ßÇÑ´Ù. HUPÀÌ³ª USR1°°Àº signalÀ» ÁÖ´Â µîÀÇ reload¸¸ ÇØ¼-´Â ¼Ò¿ëÀÌ ¾ø´Ù.)
```

¼³Á¤ (Configuration)

PHP3¸¦ ¼³Á¤ÇÏ´Â ¹æ¹ý¿¡´Â Å©°Ô µÎ°¡Áö¸¦ ÀÖ´Ù.

- PHP3¿Í ÇÔ²² Á¦°øµÇ´Â "setup" ½ºÅ©¸¦¦ »ç¿ëÇÏ´Â ¹æ¹ýÀÌ ÀÖ´Ù. ÀÌ ½ºÅ©¸Æ®´Â (PHP/FI 2.0ÀÇ "install" ½ºÅ©¸Æ®¿Í ¸Å¿ì ºñ½ÁÇÏ°Ô) ÀÏ·ÃÀÇ Áú¹®À» ÇÑ ÈÄ¿¡, ¸¶Áö¸·À¸·Î "configure"¸¦ ½ÇÇà ½ÃÅ²´Ù. ÀÌ ½ºÅ©¸¦¦ ½ÇÇà½Ã Å°·Á¸é **./setup**À» ÀÛ·ÂÇÏ¸é µÈ´Ù.

ÀÌ ½ºÅ©¸³Æ®´Â "do-conf"¶ó ºÒ¸®´Â ÆÄÀÏÀ» ¸¸µå´Âµ¥, ÀÌ ÆÄÀÏÀº "configure"¿¡°Ô ³Ñ°ÜÁÙ ¿É¼Ç »çÇ×µé¿¡ ´ëÇÑ Á¤º¸¸¦ °¡Áö°í ÀÖ´Ù. ¸¸¾à ¿©·¯°³ÐÀÌ ´ÙÁö ÇÑ,µÎ°³ÀÇ ¿É¼ÇÀ» ¼öÁ¤ÇÏ·Á ÇÑ´Ù¸é, "setup"À» ½ÇÇà½ÅÅ³ ÇÊ¿ä¾øÀÌ ÀÌ ÆÄÀÏÀ» ¼öÁ¤ÇÏ¸é µÉ °ÍÀÌ´Ù. ¼öÁ¤ÈÄ¿¡ ´Â **./do-conf**¸¦ ÀÛ·ÂÇÏ¿© »õ ¿É¼ÇÀ» °¡Áö°í configure¸¦ ½ÇÇàÇÏ¸é µÈ´Ù. (¿ªÀÚÁÖ: PHP3ÀÇ ¼³Á¤À° ¿øÄ¢ÀûÀ¸·Î configureÇÁ·Î±×·¥À¸·Î ÇÑ´Ù. setupÀº do-conf¸¦ ¸¸µé¾î configure¸¦ ½ÇÇàÇÏ±â À§ÇÑ script¿¡ ºÒ°úÇÏ´Ù.)

- ¿©·¯ºÐÀÌ ¼ÕÀ¸·Î ¼³Á¤ÇØÁÙ °æ¿ì´Â **./configure --help**¸¦ ÀÛ·ÂÇØ º¸¾Æ ¾î¶² ¿É¼ÇµéÀÌ °¡´ÉÇÑÁö ¿ì¼± »ìÆì º¸µµ ·Ï ÇÏÀÚ.

´ÙÀ½À° ¿©·¯°¡Áö ¼³Á¤ ¿É¼ÇµéÀÇ ÀÚ¼¼ÇÑ ³»¿ëÀÌ´Ù.

Apache ¸ðµâ

PHP3¸¦ ¾Æ¦ÄÄ¡ ¸ðµâ·Î ¸¸µé·Á¸é, "Build as an Apache module?"ÀÇ Áú¹®¿¡ "yes"·Î ´äÇÏ°í, Apache ¹èÆ÷º»ÀÇ base µð·º°Åä¸®¸¦ ¸í½ÃÇØ ÁÖ¸é µÈ´Ù. (configure¿¡¼´Â --with-apache=*DIR* ¿É¼ÇÀ» ÁÖ¸é µÈ´Ù.) ¸¸¾à ¿©·¯°ÐÀÇ Apache ¼ÒºÐÀÇ ¹èÆ÷º»À» /usr/local/src/apache_1.3.3¿¡ Ç®¾î ³õ¾Ò´Ù¸é, ÀÌ µð·º°Åä¸®¡¯ Apache ¹èÆ÷º»ÀÇ base µð·º°Åä¸®¡¯ µÈ´Ù. ±â°» µð·º°Åä¸®´Â /usr/local/etc/httpdÀÌ´Ù.

---

fhttpd ¸ðµâ

PHP3¸¦ fhttpd ¸ðµâ·Î ¸¸µé·Á¸é, "Build as an fhttpd module?"ÀÇ Áú¹®¿¡ "yes"·Î ´äÇÏ°í, fhttpd ¼ÒºÐÀÇ base µð·º°Åä¸®¸¦ ¸í½ÃÇØ ÁÖ¸é µÈ´Ù. (configure¿¡¼´Â --with-fhttpd=*DIR* ¿É¼ÇÀ» ÁÖ¸é µÈ´Ù.) ±â°» µð·º°Åä¸®´Â /usr/local/src/fhttpdÀÌ´Ù. ¿©·¯°ÐÀÌ fhttpd¸¦ ¿î¿µÁß¿¡ ÀÖ´Ù¸é, ÀÌ ¸ðµâ·Î ¸¸µé¾î ½ÇÇàÇÏ´Â °ÍÀÌ ´õ ÁÁÀº ¼º´ÉÀ» Á¦°øÇÏ¸ç, ¶ÇÇÑ Á¦¾îÍ¿Í ¿ø°Ý ½ÇÇà ´É·Âµµ Çâ»óµÈ´Ù.

CGI version

PHP3´Â ±â°»ÀûÀ¸·Î CGI ÇÁ·Î±×·¥À¸·Î ¸¸µé¾î Áø´Ù. ¸¸¾à ¿©·¯°ÐÀÌ PHP3°¡ ¸ðµâ·Î Á¦°øµÇ´Â À¥ ¼¹ö¸¦ ¸¸µé¸é, ÇØ´ç ¸ðµâ·Î »ç¿ëÇÏ´Â °ÍÀÌ ÀÏ¹ÝÀûÀ¸·Î ÁÁÀº ¼º´ÉÀ» ³¾ ¼ö ÀÖ´Ù. ÇÏÁö¸¸, CGI ¹öÀüÀº ¾ÆÆÄÄ¡ »ç¿ëÀÚµé¿¡°Ô ´Ù¸¥ user-id¸¦ »ç¿ëÇÏ¿© °°µµÀÇ PHP3-enabled ÆäÀÌÁö¸¦ ½ÇÇàÇÒ ¼ö ÀÖµµ·Ï ÇØ ÁØ´Ù. ¸¸¾à PHP¸¦ CGI·Î ½ÇÇàÇÏ°í ½Í´Ù¸é Security chapter¸¦ ¹Ýµå½Ã ÀÐ¾î°¡±â ¹Ù¶õ´Ù.

Database Áö¿ø ¿É¼Ç

PHP3´Â ´ÙÀ½ÀÇ databaseµé¿¡ ´ëÇÑ °íÀ¯ÀÇ Áö¿øÀ» Á¦°øÇÑ´Ù. (ODBCµµ Á¦°øÇÑ´Ù.)

Adabas D

```
--with-adabas=DIR
```

Adabas D Áö¿øÀ¸·Î ÄÄÆÄÀÏÇÑ´Ù. *DIR*Àº Adabas D°¡ ¼³Ä¡µÈ DirectoryÀÌ°í, ±â°»°ªÀº /usr/local/adabasdÀÌ´Ù.

Adabas home page

---

dBase

```
--with-dbase
```

DBaseÁö¿øÀ¸·Î ÄÄÆÄÀÏ ÇÑ´Ù. °°µµÀÇ ¶óÀÌºê·¯¸®´Â ÇÊ¿ä ¾ø´Ù.

---

filePro

```
--with-filepro
```

¹øµéµÈ ÀÐ±â Àü¿ë filePro(bundled read-only filePro)¸¦ Áö¿øÇÏµµ·Ï ÇØ ÁØ´Ù. °°µµÀÇ ¶óÀÌºê·¯¸®´Â ÇÊ¿ä ¾ø´Ù.

---

mSQL

```
--with-msql=DIR
```

mSQLÀ» Áö¿øÇÏµµ·Ï ÇÑ´Ù. *DIR*Àº mSQLÀÌ ¼³Ä¡µÈ ¸ð·º°Åä¸®ÀÌ°í ±â°»°ªÀº /usr/local/HughesÀÌ´Ù. ÀÌ ¸ð·º°Åä¸®´Â mSQL 2.0 ¹èÆ÷º»ÀÇ ±â°» ¸ð·º°Åä¸®ÀÌ´Ù. **configure**´Â ÇöÀç ÀÛµ¿ÁßÀÎ mSQLÀÇ ¹öÀüÀ» ÀÚµ¿À¸·Î ½Å·ÁÖÇÏ¿© 1.0°ú 2.0Áß ÇÏ³ª¸¦ Áö¿ø ÇÏµµ·Ï ÇÑ´Ù. ¸¸¾à PHP3°¡ mSQL 1.0Áö¿øÀ¸·Î ÄÄÆÄÀÏµÇ¾ú´Ù¸é, ¿©·¯°ÐÀº mSQL 2.0 database´Â »ç¿ëÇÏÁö ¸øÇÑ´Ù. ¸¶Áö¸·¡Á¸·Î 2.0À¸·Î ÄÄÆÄÀÏ µÇ¾ú´Ù¸é 1.0 database´Â »ç¿ëÇÒ ¼ö ¾ø´Ù.

See also mSQL Configuration Directives in the configuration file.

mSQL home page

MySQL

```
--with-mysql=DIR
```

MySQLÀ» Áö¿øÇÏµµ·Ï ÇÑ´Ù. *DIR*À° MySQLÀÌ ¼³Ä¡µÈ µð·º¸ä¸®ÀÌ°í, ±â°»°ªÀº /usr/localÀÌ´Ù. ÀÌ µð·º¸ä¸®´Â MySQL ¹èÆ÷»ÇÏ±â°» µð·º¸ä¸® ÀÌ´Ù.

See also MySQL Configuration Directives in the configuration file.

MySQL home page

---

iODBC

```
--with-iodbc=DIR
```

iODBCÁö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. ÀÌ ±â´ÉÀº iODBC µå¶óÀÌºê·ÎºÎÅÍ §ÇØ¸Ç¸Þ´º¾ó, ÀÌ iODBC µå¶óÀÌºê·ÎºÎÅÍ ¸Þ´º¾ó UNIXÇÏ¿¡¼- µ¹¾Æ°¡´Â ¹«·á ¹èÆ÷°¡ °¡´ÉÇÑ ODBC µå¶óÀÌºê·ÎºÎÅÍ´Ù. *DIR*À° iODBC°¡ ¼³Ä¡µÈ µð·º¸ä¸®ÀÌ°í, ±â°»°ªÀº /usr/localÀÌ´Ù.

FreeODBC home page

OpenLink ODBC

```
--with-openlink=DIR
```

OpenLink ODBC Áö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *DIR*À° OpenLink ODBC°¡ ¼³Ä¡µÈ µð·º¸ä¸®ÀÌ°í, ±â°»°ªÀº /usr/local/openlinkÀÌ´Ù.

OpenLink Software's home page

Oracle

```
--with-oracle=DIR
```

¿À¶óÅ¬ Áö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. ¿À¶óÅ¬ ¹öÀü 7.0¿¡¼- 7.3±îÁö´Â µ¿ÀÛÀ» Å×½ºÆ®ÇÏ¿© ¹®Á¦°¡ ¾ø¾ú´Ù. *DIR*À° ORACLE_HOME µð·º¸ä¸®ÀÌ´Ù. ¿À¶óÅ¬ È¯°æÀÌ ÀÌ¹Ì ¼³Á¤µÇ¾î ÀÖ´Ù¸é, *DIR*À» ¹Ýµå½Ã Á¤ÇØÁÙ ÇÊ¿ä´Â ¾ø´Ù.

Oracle home page

PostgreSQL

```
--with-pgsql=DIR
```

IPostgreSQLÁö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *DIR*À° PostgreSQLÀÌ ¼³Ä¡µÈ base µð·º¸ä¸®ÀÌ°í, ±â°»°ªÀº /usr/local/pgsqlÀÌ´Ù.

See also Postgres Configuration Directives in the configuration file.

PostgreSQL home page

Solid

```
--with-solid=DIR
```

SolidÁö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *=DIR*À° Solid°¡ ¼³Ä¡µÈ µð·º¸ä¸®ÀÌ°í, ±â°»°ªÀº /usr/local/solidÀÌ´Ù.

Solid home page

Sybase

```
--with-sybase=DIR
```

SybaseÁö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *DIR*À° Sybase°¡ ¼³Ä¡µÈ µð·º¸ä¸®ÀÌ°í, ±â°»°ªÀº /home/sybaseÀÌ´Ù.

See also Sybase Configuration Directives in the configuration file.

Sybase home page

Sybase - CT

```
--with-sybase-ct=DIR
```

Sybase - CTÁö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *DIR*À° Sybase - CT°¡ ¼³Ä¡µÈ µð·°Åä¸®ÀÌ°í, ±â°»°ªÀ° /home/sybaseÀÌ´Ù.

See also Sybase - CT Configuration Directives in the configuration file.

Velocis

```
--with-velocis=DIR
```

VelocisÁö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *DIR*À° Velocis°¡ ¼³Ä¡µÈ µð·°Åä¸®ÀÌ°í, ±â°»°ªÀ° /usr/local/velocisÀÌ´Ù.

Velocis home page

A custom ODBC library

```
--with-custom-odbc=DIR
```

»ç¿ëÀÚ ÁöÁ¤ÀÇ ODBC ¶óÀÌºê·¯¸® Áö¿øÀ» Æ÷ÇÔÇÏµµ·Ï ÇÑ´Ù. *DIR*À° base µð·°Åä¸®ÀÌ°í, ±â°»°ªÀ° /usr/localÀÌ´Ù.

ÀÌ ¿É¼ÇÀ» »ç¿ëÇÏ·Á¸é ¿©·¯°ÐÀ° configure ½º¼©¸®¸¦ ½ÇÇàÇÒ ¶§ CUSTOM_ODBC_LIBS¸¦ ÀÌ¹Ì Á¤ÀÇÇØ ÁÖ¾ú¾î¾ß ÇÑ´Ù. ¶ÇÇÑ, ¿©·¯°ÐÀÇ include path¿¡´Â ÀûÇÔÇÑ odbc.h¶ó´Â Çì´õ ÆÄÀÏÀÌ ¹Ýµå½Ã ÀÖ¾î¾ß ÇÑ´Ù. ¸¸¾à ¾ø´Ù¸é, ¿©·¯°ÐÀÌ ¸¸µé¾î-Æ÷ÇÔ½ÃÄÑ¾ß ÇÑ´Ù. Çì´õ¿¡´Â Á» Æ¯°°ÇÑ Á¤ÀÇµéÀÌ ÇÊ¿äÇÏ°Ô µÇ´Âµ¥, Æ¯È÷ ±×°ÍÀÌ multiplatformÀÏ °æ¿ì´Â »ó¼¼È÷ Àû¾îÁÖ¾î¾ß ÇÑ´Ù. ±× Á¤ÀÇ´ÂCFLAGS ¿É¼Ç¿¡ Àû¾îÁØ´Ù.

¿¹¸¦µé¾î ´ÙÀ½°ú °°Àº ¿É¼ÇÀ¸·Î QNX ÇÏ¿¡¼ Sybase SQL Anywhere¸¦ »ç¿ëÇÒ¼ö ÀÖ´Ù. : `CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc" ./configure --with-custom-odbc=/usr/lib/sqlany50`

Unified ODBC

```
--disable-unified-odbc
```

Unified ODBC ¸ðµâÀ» »ç¿ëÇÏÁö ¾Êµµ·ÏÇÑ´Ù. ÀÌ Unified ODBC ¸ðµâÀº Solid³ª Adabas D°°Àº ODBC±â ¹ÝÀÇ ÀÎÅÍÆäÀÌ½º (ODBC - based interface)¸¦ °¡Áö°í ÀÖ´Â ¸ðµç databaseÀÇ °øÅëÀÇ interfaceÀÌ´Ù. ¶ÇÇÑ ÀÌ ¸ðµâÀº °¸ËèÀÇ ODBC ¶óÀÌºê·¯¸®¸¦ À§ÇØ µ¿ÀÛÇÑ´Ù. ÀÌ ¿É¼ÇÀº iODBC¿Í Solid, Adabas D, Sybase SQL Anywhere¿¡¼ ½ÃÇèÀ» °ÅÃÆ´Ù. ÀÌ ¿É¼ÇÀ» »ç¿ëÇÏ·Á ¸é ÀÐÀÇ ¸ðµâµé Áß Ç³ª(¹Ýµå½Ã Ç³¸¸)°Ð³ª, vELOCIS ¸ðµâÀÌ enabledµÇ¾î ÀÖ°Ð³ª custom ODBC library°¡ ¼³Á¤µÇ¾î ÅÖ¾î¾ß ÇÑ´Ù. Áï, ÀÌ ¿É¼ÇÀº ´ÙÀ½°ú °°Àº ¿É¼ÇÁß Ç³ª°¡ »ç¿ëµÇ¾úÀ» °æ¿ì¸¸ »ç¿ëÀÌ °¡´ÉÏ´Ù. : --with-iodbc, --with-solid, --with-adabas, --with-velocis, --with-custom-odbc.

See also Unified ODBC Configuration Directives in the configuration file.

LDAP

```
--with-ldap=DIR
```

LDAP (Lightweight Directory Access Protocol) Áö¿øÀ» Æ÷ÇÔÇÑ´Ù. *DIR*À° LDAP°¡ ¼³Ä¡µÈ base µð·°Åä¸®ÀÌ°í, ±â°»°ªÀ° /usr/local/ldapÀÌ´Ù.

LDAP¿¡ ´ëÇÑ ÀÚ¼¼ÇÑ ¼³¸íÀº RFC1777 °ú RFC1778¿¡¼ Ã£À» ¼ö ÀÖ´Ù.

±âÅ¸ ¼³Á¤ ¿É¼Ç

--with-mcrypt=DIR

```
--with-mcrypt
```

mcrypt ¶óÀÌºê·¯¸®¸¦ Æ÷ÇÔÇÑ´Ù. ÀÚ¼¼ÇÑ ³»¿ëÀº mcrypt documentationÀ» °¸ÀÚ. *DIR* ¿É¼ÇÀ» ÁÖ°ÔµÇ¸é PHP´Â *DIR*/include¿¡¼ mcrypt.h¸¦ Ã£´Â´Ù.

--enable-sysvsem

```
--enable-sysvsem
```

´ë°ÎºÐÀÇ UNIX¿¡¼ Áö¿øÇÏ´Â Sys V ¼¼¸¶¦Æ÷¼¼ÎÀÇ Áö¿øÀ» Æ÷ÇÔÇÑ´Ù. ÀÚ¼¼ÇÑ Á¤º¸´Â Semaphore and Shared Memory documentationÀ» ÀÐ¾î °¸µÏ ÇÑ´Ù.

- - enable - sysvshm

```
--enable-sysvshm
```

´ë°Î°ÐÀÇ UNIX¿¡¼- Áö¿øÇÏ´Â Sys V ÀÇ °øÀ¯ ¸Þ¸ð¸®(shared memory) Áö¿øÀ» Æ÷ÇÔÇÑ´Ù. ÀÚ¼¼ÇÑ Á¤º¸´Â Semaphore and Shared Memory documentationÀ» ÀÐ¾î º¸µµ·Ï ÇÑ´Ù.

- - with - xml

```
--with-xml
```

James ClarkÀÇ expat library¸¦ »ç¿ëÇÑ, ¾ÆÁ÷ °øÀÎ¹ÞÁö ¸øÇÑ XML parser¸¦ »ç¿ëÇÏµµ·Ï ÇÑ´Ù.

ÀÚ¼¼ÇÑ ³»¿ëÀº XML function referenceÀ» º¸ÀÚ.

- - enable - maintainer - mode

```
--enable-maintainer-mode
```

PHP3°³¹ßÀÚµéÀ» À§ÇÑ ÄÄÆÄÀÏ·¯ ÁÖÀÇ ¸Þ½ÃÁö³ª, extra dependencie ¸Þ½ÃÁöµîÀ» Ãâ·ÂÇÏ°Ô ÇÑ´Ù.

- - with - system - regex

```
--with-system-regex
```

Á¤±Ô Ç¥Çö½Ä(regular expression) ¶óÀÌºê·¯¸®¸¦ PHP¿Í °°ÀÌ Á¦°øÈ °ÍÀÌ ¾Æ´Ñ, ½Ã½ºÅÛÀÇ °ÍÀ» »ç¿ëÇÏµµ·Ï ÇÑ´Ù. ¸¸¾à ¿©·¯°ÐÀÌ PHP3¸¦ ¼-¹ö¸ðµâ·Î ¸¸µç´Ù¸é, ¿©·¯°ÐÀº ¼-¹ö¸¦ ¸¸µé ¶§¿Í °°Àº ¶óÀÌºê·¯¸®¸¦ PHP3¸¦ ¸¸µé ¶§µµ »ç¿ëÇÏ¿©¾ß ÇÑ´Ù. ¸¸¾à ½Ã½ºÅÛÀÇ ¶óÀÌºê·¯¸®°¡ ¿©·¯°ÐÀÌ ¿øÇÏ´Â Æ¯°°ÇÑ ±â´ÉÀ» °¡Áö°í ÀÖ´Ù¸é ÀÌ ¿É¼ÇÀ» Enable½ÃÅ°µµ·Ï ÇÏÀÚ. ±×·¯³ª, °¡´ÉÇÏ¸é PHP¿Í °°ÀÌ Á¦øÈ ¶óÀÌºê·¯¸®¸¦ »ç¿ëÇÏ´Â °ÍÀÌ ±ÇÀåµÈ´Ù.

- - with - config - file - path

```
--with-config-file-path=DIR
```

PHP°¡ ½ÃÀÛµÉ ¶§ »ç¿ëÇÒ the php3.ini ÆÄÀÏÀ» Ã£À» µð·°Åä¸®¸¦ Á¤ÇØÁØ´Ù.

- - with - exec - dir

```
--with-exec-dir=DIR
```

¾ÈÀü ¸ðµå(safe mode)ÀÏ ¶§, ¿©±â¼- ÁöÁ¤ÇØ ÁØ *=DIR*¾ÈÀÇ ½ÇÇà ÆÄÀÏ¸¸ ½ÇÇàÀÌ °¡´ÉÇÏ´Ù. ±â°»°³À° /usr/local/binÀÌ´Ù. ÀÌ ¿É¼ÇÀº ´ÜÁö Default°³À» ÁöÁ¤ÇØ ÁØ´Â °Í »óÀÌ°í, configuration fileÀÇ safe_mode_exec_dir Áö½ÃÀÚ¿¡ ÀÇÇØ¼- ³³ÁßÇ¡ ¹Ù²ð ¼ö ÀÖ´Ù.

- - disable - debug

```
--disable-debug
```

¶óÀÌºê·¯¸®³³ ½ÇÇà ÆÄÀÏ¿¡ µð¹ö±× Á¤º¸¸¦ Æ÷ÇÔ½ÃÅ°Áö ¾Ê´Â´Ù. µð¹ö±× Á¤º¸´Â ¹ö±×ÀÇ À§Ä¡¸¦ Á¤È®ÇÏ°Ô ³³Å¸³»´Â¥ µµ¿òÀ» ÁØ´Ù. µû¶ó¼-, PHP3°¡ alpha³ª beta»óÅÂÀÌ °æ¿ì¿´Â µð¹ö±× Á¤º¸¸¦ Æ÷ÇÔ½ÃÅ°´Â °ÍÀÌ ÁÁ´Ù.

- - enable - safe - mode

```
--enable-safe-mode
```

±â°»°³À¸·Î "¾ÈÀü ¸ðµå"(safe mode)¸¦ ÁöÁ¤ÇÑ´Ù. ¾ÈÀü ¸ðµå´Â ¹®¼-ÀÇ root¿¡ ÀÖ´Â ÆÄÀÏ¸¸À» ¿-¼ö ÀÖ°Ô ÇÏ´Â µî, PHP°¡ ÇÒ ¼ö ÀÖ´Â ÀÛ¾÷¿¡ ¿©·¯ °¡Áö Á¦¾àÀ» °¡ÇÑ´Ù. ´õ ÀÚ¼¼ÇÑ Á¤º¸´Â Security chapter¸¦ ÀÐ¾î¸¸±â ¹Ù¶õ´Ù. CGI»ç¿ëÀÚµéÀº ¾ðÁ¦³ª secure mode¿¡¼- »ç¿ëÇÏ¿©¾ß ÇÑ´Ù. ÀÌ ¿É¼ÇÀº ´ÜÁö Default °³À» Á¤ÇÏ´Â °Í »óÀÌ°í, ÈÄ¿¡ configuration fileÀÇ safe_mode Áö½ÃÀÚ¿¡ ÀÇÇØ Enable/Disable µÉ ¼ö ÀÖ´Ù.

- - enable - track - vars

```
--enable-track-vars
```

HTTP_GET_VARS, HTTP_POST_VARS, HTTP_COOKIE_VARS ¹è¿-¿¡ µé¾îÀÖ´Â GET / POST / cookie °¯¼öµéÀÌ ¾îµð¼-¿Ô´ÂÁö ±â¾ïÇÏ°í ÀÖ°Ô ¸¸µç´Ù. ÀÌ ¿É¼ÇÀº ´ÜÁö Default °³À» Á¤ÇÏ´Â °Í »óÀÌ°í, ÈÄ¿¡ configuration fileÀÇ track_vars Áö ½ÃÀÚ¿¡ ÀÇÇØ Enable/Disable µÉ ¼ö ÀÖ´Ù.

- - enable - magic - quotes

```
--enable-magic-quotes
```

magic quotesÀÇ ±â°»°ªÀ» Enable·Î ÇÑ´Ù. ÀÌ ¿É¼ÇÀº ´ÜÁö Default °ªÀ» Á¤ÇÏ´Â °Í »ÓÀÌ°í, ÈÄ¿¡ configuration fileÀÇ magic_quotes_runtimeÁö½ÃÀÚ¿¡ ÀÇÇØ Enable/Disable µÉ ¼ö ÀÖ´Ù.

See also the magic_quotes_gpc and the magic_quotes_sybase directives.

- - enable - debugger

```
--enable-debugger
```

³»ÀåµÈ PHP3 µð¹ö°Å Áö¿øÀ» °¡´ÉÇÏ°Ô ÇÑ´Ù. ÀÌ ±â´ÉÀº ¾ÆÁ÷ ½ÇÇèÀûÀÎ »óÅÂÀÌ´Ù.

See also the Debugger Configuration directives in the configuration file.

- - enable - discard - path

```
--enable-discard-path
```

¸¸¾à ÀÌ°ÍÀÌ Enabledµç¸é, PHP CGI ½ÇÇàÆÄÀÏÀÌ À¥ Æ®¸® ¹Û¿¡ ¾ÈÀüÇÏ°Ô À§Ä¡ÇÒ ¼ö ÀÖ°Ô µÇ¾î »ç¿ëÀÚµéÀÌ .htaccess security¸¦ ÇÇÇÒ ¼ö ¾ø°Ô µÈ´Ù. ÀÚ¼¼ÇÑ »ç×À° section in the security chapter¸¦ ÀÐ¾î°¸±â ¹Ù¶õ´Ù.

- - enable - bcmath

```
--enable-bcmath
```

bc Çü½ÄÀÇ ¼öÀÇ Á¤¹Ðµµ(precision)¸¦ ¸¶À½´ë·Î Á¶Á¤ÇÒ ¼ö ÀÖ´Â ¼öÇÐ ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼ö ÀÖ°Ô ÇØ ÁØ´Ù.

See also the bcmath.scale option in the configuration file.

- - enable - force - cgi - redirect

```
--enable-force-cgi-redirect
```

³»°ÎÀûÀÎ ¼-¹ö ¸®µð·°Æ®(internal server redirects) ½Ã¿¡ °¸¾È °Ë»ç(security check)¸¦ ÇÏµµ·Ï ÇÑ´Ù. ¿©·¯°ÐÀÌ Apache¿Í ÇÔ²² CGI ¹öÀüÀ» »ç¿ëÇÑ´Ù¸é ¹Ýµå½Ã ÀÌ ¿É¼ÇÀ» »ç¿ëÇÏ¿©¾ß ÇÑ´Ù.

GI binaryÇüÅÂ·Î PHP¸¦ »ç¿ëÇÒ ¶§, PHP´Â ±â°»ÀûÀ¸·Î ±×°ÍÀÌ ¸®µð·°¼ÇÀ¸·Î »ç¿ëµÇ¾ú´Â°¡¸¦ ¿ì¼± °Ë»çÇÑ´Ù(¿¹¸¦ µé¾î, Apache¿¡¼- Action directives¸¦ »ç¿ëÇÑ °æ¿ìÀÌ´Ù). ÀÌ ¿É¼ÇÀ» »ç¿ëÇÏ¸é http://my.host/cgi-bin/php/secret/doc.html°ú °°Àº ¹æ¹ýÀ¸·Î PHP binary¸¦ Á÷Á¢ È£ÃâÇÏ¿© Ç¥ÁØ web server authentication ÀýÂ÷¸¦ È¸ÇÇÇÏ´Â ¹æ¹ýÀ» »ç¿ëÇÒ ¼ö ¾ø°Ô µÈ´Ù. ÀÌ ¿¹´Â http://my.host/secret/doc.html¿¡ Á¢±ÙÇÒ ¼ö ÀÖÁö¸¸, httpd°¡ /secretµð·°Åä¸®¿¡ ¼³Á¤ÇÑ ¾î¶°ÇÑ °¸È ¼³Á¤¿¡µµ ¿µÇâÀ» ¹Þ Áö ¾Ê°Ô µÈ´Ù.

ÀÌ ¿É¼ÇÀ» EnableÇÏÁö ¾ÊÀ¸¸é httpddÀÇ °¸È°ú ÀÎÁõ ¼³Á¤À» Ã¼Å©ÇÏÁö ¾Ê°í, È¸ÇÇ°¡ °¡´ÉÇÏ°Ô µÈ´Ù. ÀÌ ¿É¼ÇÀº ¼-¹ö ¼ÒÇÁ Æ®¿þ¾î°¡ ¾ÈÀüÇÑ ¸®µð·°¼ÇÀÌ µÇ¾î ÀÖ´Ù´Â °ÍÀ» ³ªÅ¸³»´Â °ÍÀÌ °Ò°¡´ÉÇÏ°í, document root¿Í »ç¿ëÀÚ µð·°Åä¸® ¾Æ·¡ÀÇ ¸ð µç ÆÄÀÏÀÌ ´©±¸¿¡°Ô³ª °³¹æµÇ¾î ÀÖÀ» ¶§¸¸ »ç¿ëÏÀÚ.

ÀÌ ¿É¼Ç¿¡ ´ëÇÑ ÀÚ¼¼ÇÑ ¾³¸¸íÀº section in the security chapterÀ» ÀÐ¾î°¸±â ¹Ù¶õ´Ù

- - disable - short - tags

```
--disable-short-tags
```

short form <? ?> PHP3 ÅÂ±×ÀÇ »ç¿ëÀ» °Ò°¡´ÉÇÏ°Ô ÇÑ´Ù. ¿©·¯°ÐÀº PHP3¿Í XMLÀ» ÇÔ²² »ç¿ëÇÒ °æ¿ì short formÀÇ »ç¿ëÀ» °Ò°¡´ÉÇÏ°Ô ÇÏ¿©¾ß ÇÑ´Ù. Â³ÀºÅÂ±×ÀÇ »ç¿ëÀÌ °Ò°¡´ÉÇÏ´Ù¸é, PHP3ÀÇ ÄÚµå´Â ÅÂ±×´Â <?php ?> »ÓÀÌ´Ù. ÀÌ ¿É¼ÇÀº ´ÜÁö Default °ªÀ» Á¤ÇÏ´Â °Í »ÓÀÌ°í, ÈÄ¿¡ configuration fileÀÇ short_open_tag Áö½ÃÀÚ¿¡ ÀÇÇØ Enable/Disable µÉ ¼ö ÀÖ´Ù.

- - enable - url - includes

```
--enable-url-includes
```

include()ÇÔ¼ö¸¦ »ç¿ëÏÒ¿© PHP3¿¡¼- Á÷Á¢ ´Ù¸¥ HTTP³ª FTP¼-¹ö¿¡ ÀÖ´Â ÄÚµå¸¦ ½ÇÇà½ÅÃ³¼ö ÀÖµµ·Ï ÇÑ´Ù.

See also the include_path option in the configuration file.

- - disable - syntax - hl

```
--disable-syntax-hl
```

syntax highlighting ±â´ÉÀ» ²ö´Ù.

CPPFLAGS and LDFLAGS

PHP3¸¦ ¼³Ä¡ÇÒ ¶§ ´Ù¸¥ µð·°Åä¸®¿¡ ÀÖ´Â Çì´õ³ª ¶óÀÌºê·¯¸®¸¦ Ã£¾Æ¼- ¸¸µé·Á¸é, CPPFLAGS ¿Í LDFLAGS È¯°æ°¯¼ö¸¦ °¢°¢ ¼öÁ¤ ÇØ ÁÖ¼î¾ß ÇÑ´Ù. ¿©·¯°³ÐÌ ÀÏ¹ÝÀûÀÎ ½©À» »ç¿ëÇÏ°í ÀÖ´Ù¸é ´ÙÀ½°ú °°ÀÌ »ç¿ëÇÏ´Â °ÍÀÌ °¡´ÉÇÒ °ÍÀÌ´Ù.
LDFLAGS= - L/my/lib/dir CPPFLAGS= - I/my/include/dir ./configure

---

Building

PHP3ÀÇ ¼³Á¤ÀÌ ¿Ï·áµÇ¸é ¿©·¯°³ÐÀº PHP3 ¶óÀÌºê·¯¸®³ª CGI ½ÇÇà ÆÄÀÏÀÀ» »ý¼ºÇÒ ÁØºñ°¡ ³¡³- °ÍÀÌ´Ù. **make** ¸í·ÉÀ¸·Î ¿©·¯°³ÐÀº ±×°ÍÀ» ¸¸µé¼ö ÀÖ´Ù. ¸¸¾à ¸¸µé¼ö ¾ø°í ±× ÀÌÀ¯¸¦ ¸ð¸¦ °æ¿ì¿¡´Â Problems sectionÀ» »ìÆìº¸¼¼â ¹Ù¶õ´Ù.

(¿ªÀÚÁÖ : PHP¸¦ Apache ¸ðµâ·Î ¸¸µé¾ú´Ù¸é restart°¡ ¾Æ´Ñ stopÈÄ startÇÏ¿©¾ß ÇÕ´Ï´Ù.)

VPATH

Testing

PHP3¸¦ CGI ÇÁ·Î±×·¥ ÇüÅÂ·Î ¸¸µé¾ú´Ù¸é, ¿©·¯°ÐÀº **make test**¶ó°í ÀÔ·ÂÇÏ¿© Å×½º°ÆÇØ º¼ ¼ö ÀÖ´Ù. ÀÌ ¹æ¹ýÀº °á°ú¹°À» Å×½º°ÆÇÏ´Â ¾ðÁ¦³ª ÁÁÀº ¹æ¹ýÀÌ´Ù. ÀÌ ¹æ¹ýÀº ¿©·¯°ÐÀÇ ½Ã½º°ÅÛ È¯°æ¿¡¼- ³ªÁß¿¡ ¹®Á¦·Î ºñ¡ ½âÀÌÁö ¾Ê°í ¹Ì¸® PHP3ÀÇ ¹® Á¦¸¦ ¹ß°ßÇÒ ¼ö ÀÖ°Ô ÇØ ÁØ´Ù.

Benchmarking

PHP3¸¦ CGI ÇÁ·Î±×·¥ ÇüÅÂ·Î ¸¸µé¾ú´Ù¸é, ¿©·¯°ÐÀº **make bench**¶ó°í ÀÔ·ÂÇÏ¿© º¥Ä¡¸¶© Å×½º°Æ®¸¦ ÇÒ ¼ö ÀÖ´Ù. ¾ÈÀüº¸ð µå(safe mode)ÀÇ ±â°»°ªÀÌ on µÇ¾î ÀÖÀ» ¶§´Â, ÀÛ¾÷ÀÌ Çã¿ëµÈ 30ÃÊ °¡´Ù ¸¹Ì ÉÀ¸®°Ô µÇ¸é º¥Ä¡¸¶©¸¦ ¾ø¾ö¾ú´Ù. ÀÌ °ÍÀº ¾ÈÀüº¸ðµå¿¡¼-´Â **set_time_limit()** ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼ö ¾øº¸ ¶§¹®ÀÌ´Ù. ½°Å©¸®Æ®¿¡¼- max_execution_timeÀ» Á¶ÀýÇÏ¿© ÀÌ ½Ã°£ °£°ÝÀ» Á¶ÀýÇÑ´Ù. **make bench**´Â configuration fileÀ» ¹«½ÃÇÑ´Ù.

PHP3 Installation Guide for Windows

ÀÌ ¼³Ä¡ ¾È³»¼-´Â ¿©·¯°Ð¿¡°Ô PHP3¸¦ Windows 9x/NT À¥¼-¹ö¿¡¼- ¼³Ä¡ÇÏ°í ¼³Á¤ÇÏ´Âµ¥ µµ¿òÀ» ÁÙ °ÍÀÔ´Ï´Ù. ÀÌ ¾È³»¼-´Â Bob Silva¿¡ ÀÇÇØ ÀÛ¼º¸µÇ¾ú½À´Ï´Ù. µû¶ó¼- °¡Åå ÃÖ½ÅÀÇ °³Á¤ÆÇÀº http://www.umesd.k12.or.us/php/win32install.html¿¡¼- Ã£À» ¼ö ÀÖÀ» °ÍÀÔ´Ï´Ù.

ÀÌ ¾È³»¼-´Â ´ÙÀ½°ú °°Àº À¥¼-¹ö¸¦ ´ë»óÀ¸·Î ÇÕ´Ï´Ù.
Personal Web Server (Newest version recommended)
Internet Information Server 3 or 4
Apache 1.3.x
Omni HTTPd 2.0b1

PHP3¿¡¼-ÀÇ ¼³Á¤ÀÇ °¯°æ

¸ðµç ¸ðµâÀº 'php3_'·Î ½ÃÀÛµÇ¾î¾ß ÇÕ´Ï´Ù. You will need to change your php3.ini file and/or any scripts loading extensions with the **dl()** function (or you could always remove the 'php3_' prefix). ÀÌ°ÍÀº php modules°ú ´Ù¸¥ ¶óÀÌ°ê·¯¸®°£ÀÇ È¥¶õÀ» À§ÇÑ °ÍÀÌ´Ù.

ChangeLog¿Í FAQ, ÃÖ½ÅÀÇ ¹®¼-µéÀº °ø½Ä PHP À¥»çÀÌÆ®¿Í ÀÌµéÀÇ ¹Ì·¯ »çÀÌÆ®¿¡¼- Ã£À» ¼ö ÀÖ½Ï´Ù.

ÀÏ¹ÝÀûÀÎ ¼³Ä¡ °úÁ¤

À¥ ¼-¹öÀÇ Á¾·ù¿Í °ü°è¾øÀÌ ´ÙÀ½ÀÇ ¼³Ä¡ °úÁ¤µéÀÌ ¿Ï·áµÇ¾î¾ß ÇÑ´Ù.

- ¹èÆ÷ ÆÄÀÏÀ» Ç®¾î ³õÀ» µð·°Åä¸®¸¦ Á¤ÇÑ´Ù. ÀÏ¹ÝÀûÀ¸·Î "C: \ PHP3 \ "µµ ÁÁ´Ù.
- 'php3 - dist.ini'ÆÄÀÏÀ» ¿©·¯¹øÇÇ '%WINDOWS%' µð·°Åä¸®¿¡ 'php3.ini'·Î °¹»çÇØ µÐ´Ù. '%WINDOWS%' µð·°Åä¸®´Â ÀÏ¹ÝÀûÀ¸·Î ´ÙÀ½°ú °°´Ù.
  c: \ windows for Windows 95/98
  c: \ winnt or c: \ winnt40 for NT servers
- 'php3.ini' ÆÄÀÏÀ» ´ÙÀ½°ú °°ÀÌ ¼öÁ¤ÇÑ´Ù.
  - 'extension_dir'À» ¿©·¯°ÐÇÇ php - install - dir·Î °íÃÄÁÖ¾î¾ß ÇÑ´Ù. È¤Àº 'php3_*.dll'ÀÌ ³õ¿©ÀÖ´Â À§Ä¡¸¦

ÁöÁ¤ÇÑ´Ù. (¿¹. c: \ php3)
- 'doc_root'´Â À¥¼-¹öÀÇ document_root µð·°Åä¸®¸¦ ÁöÁ¤ÇÏµµ·Ï ÇÕ´Ï´Ù. (¿¹.c: \ apache \ htdocs È¤À° c: \ webroot)
- PHP°¡ ½ÃÀÛµÉ ¶§ loadÇÒ ¸ðµâµéÀ» ¼±ÅÃÇÑ´Ù. 'extension=php3_*.dll' ¼³Á¤À» uncommentÇÏ¸é ÇØ´ç ¸ðµâ» loadÇÏ´Â °ÍÀÛ´Ï´Ù. ÀÏ°Î ¸ðµâÀº ¿Ã¹Ù¸£°Ô µ¿ÀÛÇÏ±â À§ÇØ ½Ã½º Û¿¡ °°µµÀÇ ¶óÀÌºê·¯¸®¸¦ ¾³Ä¡ÇØ¾ß ÇÏ´Â °æ¿ì¸µ ÀÖ´Ù. ¾îµð¼- Áö¿øÇÏ´Â ¶óÀÌºê·¯¸®¸¦ ¾òÀ» ¼ö ÀÖ´ÂÁö´Â PHPÀÇ FAQ¸¦ °¸¸é Á»´õ ÀÚ¼¼ÇÑ Á¤¸¸¦ ¾òÀ» ¼ö ÀÖÀ» °ÍÀÌ´Ù. ¿©·¯ºÐÀº ¸ðµâÀ» dl("php_*.dll"); Ã³·³ ½°Å©¸³Æ® ³»¿¼- µ¿ÀûÀ¸·Î loadÇÒ ¼öµµ ÀÖ´Ù.
- PWS¿Í IIS¿¡¼- browscap.ini´Â ´ÙÀ½ÀÇ À§Ä¡¿¡ ÀÖ´Ù. Windows 95/98À° 'c: \ windows \ system \ inetsrv \ browscap.ini', NT ¼-¹ö¿¡¼-´Â 'c: \ winnt \ system32 \ inetsrv \ browscap.ini'¿¡ ÀÖ´Ù. PHP¿¡¼- browscap±â´ÉÀ» »ç¿ëÇÏ´Â ¹æ¹ý¿¡ ´ëÇÑ Ãß°¡ÀûÀÎ ¼³¸íÀº ÀÌ mirror¿¡¼- "source" ¹öÆ°À» ¼±ÅÃÇÏ¿© °¸µµ·Ï ÇÑ´Ù.

Windows 95/98/NT¿Í PWS/IIS 3

ÀÌ ¼-¹öµé¿¡¼-ÀÇ ¼³Á¤À° ¹èÆ÷ÆÇÀÇ INF ÆÄÀÏ(php_iis_reg.inf)À» »ç¿ëÇÏ´Â °ÍÀ» ±ÇÑ´Ù. ¿©·¯ºÐÀ° ÀÌ ÆÄÀÏÀ» ¼öÁ¤ÇÏ¿© ¿©·¯ºÐÀÌ ¿øÇÏ´Â PHP ¼³Ä¡ µð·°Åä¸®¿Í È®Àå ³»¿ëµéÀ» ¼³Á¤ÇÒ ¼ö ÀÖ´Ù. ¸¸¾à ¼öµ¿Àº·Î ¼³Á¤ÇÏ´í ½Í´Ù¸é ´ÙÀ½°ú °°ÀÌ ÇÕ´Ï´Ù.

ÁÖÀÇ : ÀÌ °úÁ¤À° À©µµ¿ìÀÇ resistry¸¦ Á÷Á¢ ´Ù·ì´Ï´Ù. ÇÏ³ªÀÇ ½Ç¼ö°¡ ¿©·¯ºÐÀÇ ½Ã½º ÀüÃ¼¸¦ °Ò¾ÈÇÑ »óÅÂ·Î ¸¸µé ¼ö ÀÖ½À´Ï´Ù. ¿ì¸®´Â ¿©·¯ºÐÀÌ registry¸¦ ÆíÁýÇÏ±â Àü¿¡ ¹Ýµå½Ã ÆíÁý ÀüÀÇ registry¸¦ ¹é¾÷ÇØ ³õÀ» °Í. PHP °³ßÆÀÀ° ¾î¶°ÇÑ registryÀÇ ÆÄ¼Õµµ Ã¥ÀÓÁöÁö ¾Ê½À´Ï´Ù. ¸¸¾à registry¿¡ ¼Õ»óÀÌ °¥ °æ¿ì OS¸¦ »õ·Î ±ò±â Àü¿¡´Â °ÎÆÄµÇÁö ¾ÊÀ» ¼öµµ ÀÖ½À´Ï´Ù.

- Regedit¸¦ ½ÇÇàÇÑ´Ù.
- ´ÙÀ½À¸·Î ÀÌµ¿ÇÑ´Ù: HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap.
- edit ¸Þ´º¸¦ ¼±ÅÃÇÏ¿© New->String Value¸¦ ¼±ÃÇÑ´Ù.
- PHP script°¡ »ç¿ëÇÒ È®ÀåÀÚ¸¦ ÀÔ·ÂÇÑ´Ù. (ex: .php3)
- »õ ¹®ÀÚ¿- °ªÀ» ´õºí Å¬¸¯ÇÏ°í value ÇÊµå¿¡ php.exeÀÇ path¸¦ Àû´Â´Ù. (ex: c:\php3\php.exe %s %s) '**%s %s**'´Â ¸Å¿ì ÁßºäÇÏ´Ù. PHP´Â ÀÌ°Í¾øÀÌ´Â µ¿ÀûÇÏÁö ¾Ê´Â´Ù.
- ¸¸¾à PHP ½°Å©¸³Æ®ÀÇ ´Ù¸¥ È®ÀåÀÚ°¡ ÀÖ´Ù¸é ±×°Íµé¿¡¼- ¹«½Ã
- ´ÙÀ½À¸·Î ÀÌµ¿ÇÑ´Ù.: HKEY_CLASSES_ROOT
- edit ¸Þ´º¿¡¼- New->Key¸¦ ¼±ÅÃÇÑ´Ù.
- ¿©·¯ºÐÀÌ ÀÌÀü °úÁ¤¿¡¼- ¾³Á¤ÇÑ È®ÀåÀÚ¸¦ KeyÀÇ ÀÌ¸§À¸·Î ÁöÁ¤ÇÑ´Ù. (ex: .php3)
- »õ Å°¸¦ Highlight½ÃÅ°°í ¿À¸¥ÂÊ Ã¢¿¡¼- "default value" ¸¦ ´õºí Å¬¸¯ÇÑ ÈÄ¿¡ phpfileÀÌ¶ó°í Àû´Â´Ù.
- À§¿¡¼- ¾³Á¤ÇÑ ´Ù¸¥ È®ÀåÀÚ°¡ ÀÖÀ¸¸é ´Ù¸¥ È®ÀåÀÚ¿¡ ´ëÇØ¼-µµ ÀÌ °úÁ¤À» ¹Ý°¹ÇÑ´Ù.
- ÀÌÁ¦ HKEY_CLASSES_ROOT¿¡¼- New->Key¸¦ ¼±ÅÃÇÏ¿© »õ·Î¿î Key¸¦ ¸¸µé°í phpfile·Î ÀÌ¸§À» Á¤ÇÑ´Ù.
- phpfileÀÌ¶ó°í Á¤ÇÑ »õ Å°¸¦ Highlight½ÃÅ°°í ¿À¸¥ÂÊ Ã¢¿¡¼- "default value" ¸¦ ´õºí Å¬¸¯ÇÑ ÈÄ¿¡ PHP Script¶ó°í Àû´Â´Ù.
- phpfile key¿¡¼- ¿À¸¥ÂÊ ¹öÆ°À» Å¬¸¯ÇÏ¿© New->Key¸¦ ¼±ÅÃÇÏ°í ShellÀÌ¶ó°í ÀÌ¸§ °ÙÀÎ´Ù.
- Shell key¿¡¼- ¿À¸¥ÂÊ ¹öÆ°À» Å¬¸¯ÇÏ¿© New->Key¸¦ ¼±ÅÃÇÏ°í openÀÌ¶ó°í ÀÌ¸§ °ÙÀÎ´Ù.
- open key¿¡¼- ¿À¸¥ÂÊ ¹öÆ°À» Å¬¸¯ÇÏ¿© New->Key¸¦ ¼±ÅÃÇÏ°í command¶ó°í ÀÌ¸§ °ÙÀÎ´Ù.
- command¶ó°í Á¤ÇÑ »õ Å°¸¦ Highlight½ÃÅ°°í ¿À¸¥ÂÊ Ã¢¿¡¼- "default value" ¸¦ ´õºí Å¬¸¯ÇÑ ÈÄ¿¡ php.exeÀÇ path¸¦ Àû´Â´Ù. (ex: c:\php3\php.exe -q %1. (%1À» ÀØÁö ¸»µµ·Ï)).
- Regedit¸¦ Á¾·áÇÑ´Ù.

PWS¿Í IIS 3 »ç¿ëÀÚµéÀ° ÇöÀç ¿ÏÀüÇÑ ¿î¿µÁßÀÎ ½Ã½º Û¿¡ ÀÖ½À´Ï´Ù. IIS 3 »ç¿ëÀÚµéÀ° Steven Genusa°¡ ÀÛ¼ºÇÑ ½°Å©¸³Æ® ¸ÊÀ» ¼³Á¤ÇÏ´Â ¸·ÀÀûÀÎ toolÀ» »ç¿ëÇÒ ¼ö ÀÖ½À´Ï´Ù.

Windows NT¿Í IIS 4

PHP3¸¦ IIS 4°¡ ¿î¿µÁßÀÎ NT Server¿¡ ¼³Ä¡ÇÏ·Á¸é ´ÙÀ½ °úÁ¤À» µû¸¥´Ù.

- Internet Service Manager (MMC)¿¡¼-, Web site¸¦ ¼±ÅÃÇÏ°Å³ª applicationÀÇ ½ÃÀÛÀÌ µÇ´Â µð·°Åä¸®¸¦ ¼±ÅÃÇÑ´Ù.
- ¿À¸¥ÂÊ ¹öÆ°À» Å¬¸¯ÇÏ°í properties(µî·ÏÁ¤¸¸)¸¦ ¼±ÅÃÇÏ¿©, µð·°Åä¸®ÀÇ property sheets(µî·Ï Á¤¸¸ È-¸é)À» ¿¬ÈÄ¿¡, Home Directory³ª Virtual Directory È¤Àº Directory ÅÇÀ» Å¬¸¯ÇÑ´Ù.
- Configuration ¹öÆ°À» Å¬¸¯ÇÑÈÄ App Mappings ÅÇÀ» Å¬¸¯ÇÑ´Ù.
- Add¸¦ Å¬¸¯ÇÏ¿© Executable ¹Ú½º¿¡¼- ´ÙÀ½°ú °°ÀÌ ÀÔ·ÂÇÑ´Ù. : c:\path-to-php-dir\php.exe %s %s. À¸¶§ ¹Ýµå½Ã ¸ÇµÚ¿¡ %s %s¸¦ °ÙÀÎÇÕ´Ï´Ù. ±×·¸Áö ¾ÊÀ¸¸é PHP´Â ¿Ã¹Ù¸£°Ô ÀÛµ¿ÇÏÁö ¾Ê½À´Ï´Ù.
- Extension ¹Ú½º¿¡¼- PHP½º¸©¸³Æ®µéÀÇ ÆÄÀÏ È®ÀåÀÚ¸¦ ÀÔ·ÂÇÕ´Ï´Ù. °¢°¢ÀÇ È®ÀåÀÚ¿¡ ´ëÇÏ¿© ÀÌ °úÁ¤À» ¹Ý°¹ ÇÕ´Ï´Ù. (.php3¿Í .phtmlÀº °øÀëÀÛ´Ï´Ù.)
- Àû´çÇÑ °¸ÈÀ» ¼³Á¤ÇÑ´Ù. (ÀÌ°ÍÀ¸·Î Internet Service Manager¿¡¼-ÀÇ ¼³Á¤À° ³¡ÀÌ´Ù.) ¸¸¾à ¿©·¯ºÐÀÌ NT Server¿¡ NTFS ÆÄÀÏ ½Ã½º ÅÛÀ» »ç¿ëÇÑ´Ù¸é, php.exe°¡ ÀÖ´Â ðð·°Åä¸®¿¡ ´ëÇÏ¿© I_USR_ »ç¿ëÀÚ¿¡°Ô ½ÇÇà ±ÇÑÀ» ÁÖ¾î¾ß ÇÑ´Ù.

Windows 9x/NT¿Í Apache 1.3.x

Apache¸¦ PHP CGI binary¿Í ÇÔ²² ½ÇÇàÇÏµµ·Ï ¼³Á¤ÇÏ·Á¸é srm.confÀÌ³ª httpd.conf¸¦ ´ÙÀ½°ú °°ÀÌ ¼öÁ¤ÇÏ¿©¾ß ÇÑ´Ù.

°ñ·Ï ApacheÇÏ¿¡¼- PHPÀÇ ¼³Á¤ÀÌ ¾à°£ Â÷ÀÌ°¡ ÀÖÀ»Áö ¸ð¸£ÁÖ¸¸, ÀÌ ¼³Á¤À° ÃÊ°¡ÀÚµµ ÇÒ ¼ö ÀÖÀ» Á¤µµ·Î ½±´Ù. ¿©±â¿¡ ³ª¿Â °Í ÀÌ»óÀÇ ¼³Á¤ Áö½ÃÀÚ´Â Apache Docs¸¦ ÂüÁ¶ÇÏµµ·Ï ÇÏÀÚ.

- `ScriptAlias /php3/ "c:/path-to-php-dir/php.exe"`
- `AddType application/x-httpd-php3 .php3`
- `AddType application/x-httpd-php3 .phtml`
- `Action application/x-httpd-php3 "php3/php.exe"`

To use the source code highlighting feature, simply create a PHP script file and stick this code in: . Substitute `original_php_script.php3` with the name of the file you wish to show the source of. (this is only one way of doing it).

Note: Win - Apache¿¡¼- "c: \ directory \ file.ext"¿Í °°ÀÌ path¿¡ »ç¿ëµÇ´Â ¸ðµç ¹é½½·¡½Ã(\ )´Â ÀÏ¹Ý ½½·¡½Ã(/)·Î ¹Ù²ã¾ß ÇÔ´Ï´Ù.

Windows¿¡¼- Omni HTTPd 2.0b1

ÀÌ ¼³Á¤À° ¸Å¿ì ½¬¿î ÆíÀÌ´Ù.
Step 1: Omni server¸¦ ¼³Ä¡ÇÑ´Ù.
Step 2: system trayÀÇ Çª¸¥»ö OmniHTTPd ¾ÆÀÌÄÜ¿¡¼- ¿À¸¥ÂÊ ¹öÆ°À» Å¬¸¯ÇÏ¿© `Properties(µî·ÏÁ¤°¸)`¸¦ ¼±ÅÃÇÑ´Ù.
Step 3: `Web Server Global Settings`À» Å¬¸¯ÇÑ´Ù.
Step 4: 'External' ÅÇ¿¡¼- ´ÙÀ½À» ÀÔ·ÂÇÑ´Ù. : `virtual = .php3 | actual = c:\path-to-php-dir\php.exe`
Step 5: `Mime` ÅÇ¿¡¼- ´ÙÀ½À» ÀÔ·ÂÇÑ´Ù. : `virtual = wwwserver/stdcgi | actual = .php3`
Step 6: `OK`¸¦ Å¬¸¯ÇÑ´Ù.

PHPÀÇ ¸ðµç È®ÀåÀÚ¿¡ ´ëÇØ steps 2 - 6À» ¹Ý°¹ÇÑ´Ù.

PHP Modules

Table 3 - 1. PHP Modules

php3_calendar.dll	Calendar conversion functions
php3_crypt.dll	Crypt functions
php3_dbase.dll	DBase functions
php3_dbm.dll	GDBM emulation via Berkely DB2 library
php3_filepro.dll	READ ONLY access to filepro databases
php3_gd.dll	GD Library functions for gif manipulation
php3_hyperwave.dll	HyperWave functions
php3_imap4r2.dll	IMAP 4 functions
php3_ldap.dll	LDAP functions
php3_msql1.dll	mSQL 1 client
php3_msql2.dll	mSQL 2 client
php3_mssql.dll	MSSQL client (requires MSSQL DB - Libraries)
php3_mysql.dll	MySQL functions
php3_nsmail.dll	Netscape mail functions
php3_oci73.dll	Oracle functions
php3_snmp.dll	SNMP get and walk functions (NT only!)
php3_zlib.dll	ZLib functions

Problems?

Read the FAQ

ÀÏ°Î Áú¹®µéÀº °øÅëÀûÀÎ °ÍÀÌ ¸¹´Ù. PHP3 FAQ¿¡ ÀÌ °øÅëÀûÀÎ Áú¹®µéÀÌ ¸ð¿©ÀÖ¸´Ï Ã£¾Æ°¸ÀÚ. ÀÌ ¹®¼-´Â http://www.php.net/FAQ.php3¿¡¼- Ã£À» ¼ö ÀÖ´Ù.

(¿ªÀÚÁÖ : FAQÀÇ ¹ø¿ª°»À° http://w3.to/regina/FAQ.htm¿¡ ¹ø¿ªµÇ¾î ÀÖ¸³ª ÃÖ½ÅÆÇÀº ¾Æ´Ï´Ù.)

Bug reports

¸¸¾à ¿©·¯°ÐÀÌ Bug¶ó°í »ý°¢µÇ´Â °ÍÀÌ ¹ß°ßµÉ °æ¿ì ÀÌ°ÍÀ» ¹Ýµå½Ã °¸°íÇØ ÁÖ±â ¹ù¹õ¿¡ ´ëÇØ ¿©·¯°ÐÀÌ ¾Ë·ÁÄÖ±â Àü¿¡ Â¸ð¸£ °ÍÀ» °ÍÀÌ´Ù. ¿©·¯°ÐÀÇ ¹ö±× ·¹Æ÷Æ®·Î ±× ¹ö±×µéÅ» ÀâÀ» ¼ö ÀÖ´Ù. ¹ö±× °¸°í¿ë ÆûÀÌ PHP3 »çÀÌÆ®¿¡ ÀÖ´Ù. ÁÖ¿ä ¾ç½ÅÀº http://ca.php.net/bugs.php3¿¡¼- Ã£À» ¼ö ÀÖÀ» °ÍÀÌ´Ù. (¿ªÀÚÁÖ: ¹ö±× ·¹Æ÷Æ®´Â ¿µ¾î·Î ½á¾ß °ÚÁö¿ä. Hi)

Other problems

¸¸¾à ¿©·¯°ÐÀÌ ¾ÆÁ÷µµ °í»ýÇÏ°í ÀÖ´Ù¸é, PHP3ÀÇ ¸ÞÏ¸µ ¸®½ºÆ®(mailing list)°¡ µµ¿òÀ» ÁÙ ¼ö ÀÖÀ» °ÍÀÌ´Ù. ¿©·¯°ÐÀÌ

¸¸¾à ¾î¶² Áú¹®À» ÇÏ±â Àü¿¡ ¿ì¼± °¸°ü ¹®¼-(archive)¸¦ Ã£¾Æ°¸Æ °°À° ¹®Á¦¿¡ ´ëÇÑ ÇØ´äÀÌ ÀÖ´ÂÁö Ã£¾Æ°¸´Â °ÍÀ» ÀØÁö ¸»µ·Ï ÇÏÀÚ. °¸°ü ¹®¼-´Â http://www.tryc.on.ca/php3.html¿¡¼- Ã£¾Æ °¼ ¼ö ÀÖ´Ù. PHP3 ¸ÞÀÏ¸µ ¸®½°Æ®¿¡ Âü°¡ÇÏ·Á¸é php3-subscribe@lists.php.netÀ¸·Î °ó ÆíÁö¸¦ ÇÏ³ª °¸³»¸é µÈ´Ù. mailing list ÁÖ¼Ò´Â php3@lists.php.netÀÌ´Ù.

¸¸¾à ¿©·¯°ÐÀÌ mailing list¸¦ ÅëÇØ¼- ¾î¶² µµ¿òÀ» ¹Þ°í ½Í´Ù¸é, Áú¹®½Ã¿¡ ÀÚ¼¼ÇÏ°í ¸íÈ®ÇÑ Áú¹®°ú ÇÔ²², ¿©·¯°ÐÀÇ ¿î¿µ È¯°æ¿¡ ´ëÇÑ ¼¼°ÎÇ×¸ñÀ» Ã·°ÎÇÏ´Â °ÍÀÌ ÁÁ´Ù (¿î¿µÃ¼°è, PHP ¹öÀü, À¥¼-¹öÀÇ Á¾·ù¿Í ¹öÀü, PHP¸¦ CGI·Î ¾²°í ÀÖ´ÂÁö ¸ð µâ·Î ¾²°í ÀÖ´ÂÁö µî). ¶ÇÑ ¿©·¯°ÐÀÌ ´ç¸éÇÑ ¹®Á¦¸¦ ÀÇÇöÇÒ ¼ö ÀÖ´Â ÄÚµå¸¦ Æ÷ÇÔ½ÃÅ°¸é ´õ¿í ÁÁ´Ù. (¿ªÀÚÁÖ: ÀÌ°Íµµ ´ç¿¬È÷ ¿µ¾î·Î ½á¾ß °ÚÁö¿ä. Hi)

Security

PHP´Â ¸Å¿ì °-·ÂÇÑ µµ±¸ÀÌ´Ù. ´Ù¸¥ ¸¹À° °-·ÂÇÑ µµ±¸¿Í °°ÀÌ, ÀÌ °-·ÂÇÑ ±â´ÉÀÌ ¿©·¯°Ð ÀÚ½Å¿¡°Ô ÇÇÇØ¸¦ ÁÙ ¼öµµ ÀÖ´Ù. PHP¸¦ ¹«½ÉÄÚ »ç¿ëÇÏ´Ù °¸¸é, ¿©·¯°ÐÀÇ ½Ã½°ÅÛ¿¡ ½É°¢ÇÑ °¸¾È ¹®Á¦¸¦ °¡Á® ÀÙ¼öµµ ÀÖ´Ù. ÀÌ·¯ÇÑ ¹®Á¦¸¦ ¹æÁöÇÏ´Â °¡Àå ÁÁÀº ¹æ¹ýÀº ¿©·¯°ÐÀÌ ÇÏ´Â ÀÛ¾÷ÀÌ ¹«¾ùÀÎÁö¸¦ È®½ÇÈ÷ ¾Ë°í ÇÏ´Â °ÍÀÌ´Ù. °¸´Ù ÀÚ¼¼ÇÑ »çÇ×À° Security chapter¸¦ ÀÐ¾î°¸ ±â ¹Ù¶õ´Ù.

Chapter 4. ¼³Á¤(Configuration)

The php3.ini file

php3.iniÆÄÀÏÀº PHPÀÇ parser°¡ ½ÇÇà µÉ ¶§ ÀÐÇôÁø´Ù. ¼-¹ö ¸ðµâ ÇüÅÂÀÇ PHP´Â ÀÌ ÀÛ¾÷À» ¼-¹ö°¡ ½ÇÇà µÉ ¶§ ÇÑ¹ø¸¸ ÇÏ°Ô µÉ´Ù. CGIÇüÅÂÀÇ °æ¡ì´Â °Ò·ÁÁú ¶§¸¶´Ù ¸Å¹ø ÀÏ¾î³´Ù.

ÀÌ°÷¿¡ ³ª¿-µÈ °³°³ÀÇ Áö½ÃÀÚ(directive)µéÀº ±×°Í°ú ÀÏÄ¡ÇÏ´Â ApacheÀÇ httpd.conf Áö½ÃÀÚ¸¦ °¡Áø´Ù. httpd.conf¿¡¼- ±× Áö½ÃÀÚ¸¦ »ç¿ëÇÏ´Â ¹æ¹ýÀº ´Ü¼øÈ÷ ÀÌ°÷¿¡ ³ª¿-µÈ ÀÌ¸§ ¾Õ¿¡ *php3_*¸¦ °¡Á® ÀÌ¸§À» ¹Ù²Ù¸é µÈ´Ù.

(¿ªÀÚÁÖ : PHP¸¦ ¼ÆÆÄ¡ ¸ðµâ·Î »ç¿ëÇÒ °æ¡ì httpd.conf¿¡ Àû¾îÁØ ¿É¼Ç¸¸ ÀÛµµ¿ÇÑ´Ù.)
(¿ªÀÚÁÖ : httpd.conf¿¡ ¿É¼ÇÀ» Àß´Â ¹æ¹ýÀº Á¶±Ý ´Ù¸£´Ù. "="À» »ç¿ëÇÏÁö »ç°³É ÇÑÄ-À» ¶ç°í Àû¾îÁØ¸é µÈ´Ù. *include_path* ¿É¼ÇÀ» ¼³Á¤ÇÏ´Â ¿¹´Â ´ÙÀ½°ú °°´Ù. php3_include_path .:/home/httpd/php-lib)

´ë°Î°ÐÀÇ ¼³Á¤°ªÀº **phpinfo()**ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© È®ÀÎÇØ º¼ ¼ö ÀÖ´Ù.

General Configuration Directives

auto_append_file string
: main ÆÄÀÏ ÈÄ¿¡ ÀÚµ¿ÀûÀ¸·Î ÇØ¼®µÉ ÆÄÀÏÀÇ ÀÌ¸§À» ¸í½ÃÇÑ´Ù. ±× ÆÄÀÏÀº **include()**ÇÔ¼ö¸¦ ÅëÇØ¼- Æ÷ÇÔµÇ¹Ç·Î include_path Áö½ÃÀÚ°¡ »ç¿ëµÈ´Ù.

  noneÀÌ¶ó´Â Æ¯°°ÇÑ °ªÀ» ÁÖ°Ô µÇ¸é auto-appendingÀÌ disableµÈ´Ù.

  NOTE: ¸¸¾à ½°Å©¸Æ®¡ exit()·Î Á¾·áµÇ°Ô µÇ¸é auto-append´Â ÀÏ¾î³ªÁö ¾Ê´Â´Ù.

  > **NOTE:** ¸¸¾à ½°Å©¸Æ®¡ **exit()**·Î Á¾·áµÇ°Ô µÇ¸é auto-append´Â ÀÏ¾î³ªÁö ¾Ê´Â´Ù.

auto_prepend_file string
: main ÆÄÀÏ Àü¿¡ ÀÚµ¿ÀûÀ¸·Î ÇØ¼®µÉ ÆÄÀÏÀÇ ÀÌ¸§À» ¸í½ÃÇÑ´Ù. ±× ÆÄÀÏÀº **include()** ÇÔ¼ö¸¦ ÅëÇØ¼- Æ÷ÇÔµÇ¹Ç·Î include_path Áö½ÃÀÚ°¡ »ç¿ëµÈ´Ù.

  noneÀÌ¶ó´Â Æ¯°°ÇÑ °ªÀ» ÁÖ°Ô µÇ¸é auto-prependingÀÌ disableµÈ´Ù.

cgi_ext string

display_errors boolean
: ÀÌ °áÁ¤ÀÚ(determine)´Â ¿¡·¯¡ »ý°åÀ» ¶§ HTML Á¶°ÇÀÇ Ãâ·ÂÀ» ½°Å©¸°À¸·Î °¸ÀÌ °ÍÀÎ°¡ ¾Æ´Ñ°¡¸¦ °áÁ¤ÇÑ´Ù.

doc_root string
: PHPÀÇ "root directory"¸¦ ÁöÁ¤ÇÑ´Ù. ¹Ýµå½Ã °ªÀ» °¡Á®¾ß ÇÑ´Ù. PHP°¡ ¾ÈÀü¸ðµå(safe mode)·Î ¾³Á¤µÇ¾î ÀÖÀ» °æ¡ì, ÀÌ µð·°Åä¸® ¹Û¿¡ ÀÖ´Â ÆÄÀÏÀº »ç¿ëÇÒ ¼ø´Ù.

engine boolean
: ÀÌ ÁöÁÀÚ´Â ¼ÆÆÄ¡ ¸ðµâ ¹öÀüÀÇ PHP¿¡¼-¸¸ À¯È¿ÇÏ´Ù. ÀÌ°ÍÀº »çÀÌÆ®¿¡¼- µð·°Åä¸® ´ÜÀ§³ª, Virtual ¼-¹ö ´ÜÀ§¸¦ ±â ¹ÝÀ¸·Î ±´ÉÀÌ PHP ÆÄ½ÌÀ» ÄÑ°Å³ª ²ô¸·Ï ÇÑ´Ù. httpd.conf ÆÄÀÏÀÇ Àû´çÇÑ À§Ä¡¿¡ **php3_engine off**¶ó°í ¸í½ÃÇÒ¸·, PHP¸¦ °¡´ÉÇÏ°Å³ª °Ò°¡´ÉÇÏ°Ô ÇÒ ¼ö ÀÖ´Ù.

error_log string

½ºÅ©¸³Æ® ¿¡·¯°¡ ·Î±×µÉ ÆÄÀÏ À̸§À» Á¤ÇÑ´Ù. ·Î±× ÆÄÀÏ·Î ½Ã½ºÅÛ ·Î±×¸¦ »ç¿ëÇÏ±â À§ÇØ¼ syslog¶ó´Â Æ¯°°ÇÑ °³ÀÌ ÀÌ ¿ëµÈ´Ù. À¯´Ð½º¿¡¼´Â syslog(3)ÀÌ°í, Windows NT¿¡¼´Â Event log°¡ µÈ´Ù. Windows 95¿¡¼´Â ½Ã½ºÅÛ ·Î±×¸¦ Áö¿øÇÏÁö ¾Ê´Â´Ù.

error_reporting integer
¿¡·¯ ·¹Æ÷ÆÃÀÇ ·¹º§À» ¼³Á¤ÇÑ´Ù. ÀÌ ÆÄ¶ó¸ÞÅ¸ °ªÀº °ñÆ® ´ÜÀ§·Î ¼³Á¤ÇÑ´Ù. ´ÙÀ½ Áß¿¡ ¿øÇÏ´Â ¿¡·¯ ·¹Æ÷ÆÃ ·¹º§À̳ª °ªÀ» ´õÇØÁÖ¸é µÈ´Ù.

Table 4 - 1. Error Reporting Levels

bit value	enabled reporting
1	normal errors
2	normal warnings
4	parser errors
8	non - critical style - related warnings

±â°»°ªÀº 7 ÀÌ´Ù(normal errors¿Í normal warnings, parser errors°¡ ·¹Æ÷Æ® µÈ´Ù).

open_basedir string
PHP°¡ ¿-¾î°¼ ¼öÀÖ´Â ÆÄÀÏ» Æ¯Á¤ µð·ºÄä¸® ÀÌÇÏ·Î Á¦ÇÑÇÑ´Ù.

¿¹¸¦ µé¾î ¾î¶² ½ºÅ©¸³Æ®°¡ fopenÀÌ³ª gzopenÀ¸·Î ÆÄÀÏÀ» ¿-·Á°í ÇÒ ¶§ ±× ÆÄÀÏÀÌ ÁöÁ¤µÈ À§Ä¡¿¡ ÀÖ´Â°¡ °Ë»çÇÑ´Ù. ¸¸¾à ÆÄÀÏÀÌ ÁöÁ¤µÈ µð·ºÆ¼¸®Ûº¡ ÀÖ´Ù¸é, PHP´Â ÀÌ ¿-±â µ¿ÀÛÀ» °ÅºÎÇÑ´Ù. ¸ðµç symbolic linkµéÀº ¿ø·¡ÀÇ ÀÌ¸§À¸·Î ¹Ù²î¾î ÇØ¼®µÈ´Ù. µû¶ó¼ ÀÌ Á¦ÇÑ» ÇÇÇÏ±â À§ÇØ symbolic link¸¦ »ç¿ëÇÏ´Â °ÍÀ» ¹æÁöÇÒ ¼ö ÀÖ´Ù.

.ÀÌ¶ó´Â ÁöÁ¤µÈ °ªÀ» ÁÖ¸é ½ºÅ©¸³Æ® ÆÄÀÏÀÌ ÀÖ´Â µð·ºÄä¸®¡ base - directory·Î ÁöÁ¤µÈ´Ù.

±â°»°ªÀº ¸ðµç ÆÄÀÏÀ» ¿- ¼ö ÀÖ´Â °ÍÀÌ´Ù.

gpc_order string
GET / POST / COOKIE °¢¼öµéÀÇ ÇØ¼® ¼ø¼¸¦ Á¤ÇÑ´Ù. ±â°»°ªÀº "GPC"ÀÌ´Ù. ¿¹¸¦µé¾î "GP"¶ó°í ¾³Á¤ÇÑ´Ù¸é, PHP´Â cookieµéÀº ¿ÏÀüÈ÷ ¹«½ÃÇÏ°í, °°Àº ÀÌ¸§À» °¡Áö POST °¢¼ö°¡ GET °¢¼ö°¡ POST °¢¼öÀÇ °ªÀ¸·Î µ¤¾î ¾²¿©Áö°Ô µÈ´Ù.

ignore_user_abort string
±â°»°ªÀº OffÀÌ´Ù. ¸¸¾à ÀÌ °ªÀÌ OnÀÌ¸é, ¿ø°Ý Å¬¶óÀ¼ðÆ®°¡ Áß°£¿¡ ¿¬°áÀ» ²÷¾îµµ ½ºÅ©¸³Æ®°¡ ³¡±îÁö ¼öÇàµÈ´Ù.

See also **ignore_user_abort()**.

include_path string
require()¿Í **include()**, **fopen_with_path()** ÇÔ¼öµéÀÌ »ç¿ëÇÏ´Â ÆÄÀÏÀ» Ã£À» À§Ä¡(path)¸¦ ¸í½ÃÇÑ´Ù. stringÀÇ Çü½ÄÀº ½Ã½ºÅÛ È¯°æ¯¼öÀÎ PATH ÀÇ ¾³Á¤°ú °ñ½ÁÇÏ´Ù. UNIX¿¡¼´Â ÄÝ·ÐÀ¸·Î ³ª´µ¾îÁø µð·ºÄä¸®µéÀÇ ¸®½ºÆ®ÀÌ°í, Windows¿¡¼´Â ¼¼¹ÌÄÝ·ÐÀ» »ç¿ëÇÑ´Ù.
Example 4 - 1. UNIX include_path

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

Example 4 - 2. Windows include_path

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

±â°»°ªÀº .ÀÌ´Ù. (ÇöÀç µð·ºÄä¸®¸¸ À» ³ªÅ¸³½´Ù)

isapi_ext string

log_errors boolean
½ºÅ©¸³Æ® ¿¡·¯°¡ ¸Þ½ÃÁö°¡ ¼-¹öÀÇ ¿¡·¯·Î ·Î±×µÇ´Â °¡¸¦ Á¤ÇÑ´Ù. ÀÌ ¿É¼ÇÀº ¼-¹ö¿¡ µû¶ó ´Ù¸£°Ô µ¿ÀÛÇÑ´Ù.

magic_quotes_gpc boolean
GPC (Get/Post/Cookie) µ¿ÀÛ Áß¿¡ ÀÏ¾î³ª´Â magic_quotes ±â´É» ¾³Á¤ÇÑ´Ù. magic_quotes°¡ ¾³Á¤µÇ¾î ÀÖ¸¸ GPC¿¡¼ ÀÛ·ÂµÈ ¸ðµç ' (single - quote)¿Í " (double quote), \ (backslash), NUL µéÀÇ ¾Õ¿¡ ¹é½½·¡½Ã(\ )°¡ ÀÚµ¿À¸·Î °ÙºÔ µÈ´Ù. ¶ÇÇÑ, magic_quotes_sybaseµµ °°ÀÌ ¾³Á¤µÇ¾î ÀÖ¸¸ '(single - quote)´Â ¹é½½·¡½Ã(\ )°¡ ¾Æ´Ñ single - quote(')°¡ ¾Õ¿¡ °ÙºÔ µÈ´Ù.

magic_quotes_runtime boolean
¸¸¾à *magic_quotes_runtime*ÀÌ ¾³Á¤µÇ¾î ÀÖ´Ù¸é, database¿Í text ÆÄÀÏ¿¡¼ ÀÐ¾î µéÀÌ´Â ¸ðµç µ¥ÀÌÅÍ¿¡ ÀÖ´Â ' (single - quote)¿Í " (double quote), \ (backslash), NUL µéÀÇ ¾Õ¿¡ ¹é½½·¡½Ã(\ )¸¦ °ÙÀÌµµ·Ï ÇÑ´Ù. ¶ÇÇÑ, magic_quotes_sybaseµµ °°ÀÌ ¾³Á¤µÇ¾î ÀÖÀ¸¸é '(single - quote)´Â ¹é½½·¡½Ã(\ )°¡ ¾Æ´Ñ single - quote(')°¡ ¾Õ¿¡ °ÙºÔ µÈ´Ù.

magic_quotes_sybase boolean
¸¸¾à *magic_quotes_gpc*³ª *magic_quotes_runtime*ÀÌ ¾³Á¤µÇ¾î ÀÖ°í, magic_quotes_sybaseµµ °°ÀÌ ¾³Á¤µÇ¾î ÀÖÀ¸¸é '(single - quote)´Â ¹é½½·¡½Ã(\ )°¡ ¾Æ´Ñ single - quote(')°¡ ¾Õ¿¡ °ÙºÔ µÈ´Ù.

max_execution_time integer
±¸¹®ÇØ¼®±â(parser)°¡ ÇÑ ½º¸®½º¸¦ Ã³¸®ÇÏ´Âµ¥ °É¸®´Â ÃÖ´ë ½Ã°£À» ¼³Á¤ÇÑ´Ù. ÀÌ°ÍÀº °Ò¿ÏÀüÇÑ ½º¸®½º¸®·Î°ÎÍ ¼-¹ö°¡ ¸ðµÎ Á¡À¯´çÇÏ´Â °ÍÀ» ¹æÁöÇÏ´Âµ¥ µµ¿òÀÌ µÈ´Ù.

memory_limit integer
ÇÑ ½º¸®½º¸®°¡ ÇÒ´ç¹Þ¾» ¼ö ÀÖ´Â ¸Þ¸ð¸®¹®ÀÇ ÃÖ´ë Å©±â¸¦ ÁöÁ¤ÇÑ´Ù. ÀÌ°ÍÀº °Ò¿ÏÀüÇÑ ½º¸®½º¸®·Î°ÎÍ ¼-¹ö ÀüÃ¼ÀÇ ¸Þ¸ð¸®°¡ ÀâÇô ¸ÔÈ÷´Â °ÍÀ» ¹æÁöÇÑ´Ù.

nsapi_ext string

short_open_tag boolean
PHPÀÇ Open ÅÂ±×·Î short form (<? ?>)ÀÇ »ç¿ëÀ» Çã°¡ÇÏ°Å³ª ºÒÇãÇÑ´Ù. ¿©·¯°ÐÀº PHP3¿Í XMLÀ» ÇÔ²² »ç¿ëÇÒ °æ¿ì short formÀÇ »ç¿ëÀ» °Ò°¡´ÉÇÏ°Ô ÇÏ¿©¾ß ÇÑ´Ù. Â³ÀºÀº ÅÂ±×ÀÇ »ç¿ëÀÌ °Ò°¡´ÉÇÏ´Ù¸é, PHPÀÇ Open ÅÂ±×·Î´Â long form (<?php ?>)¸¸À» »ç¿ëÇÏ¿©¾ß ÇÑ´Ù.

sql.safe_mode boolean

track_errors boolean
ÀÌ ÁöÌÃÀÚ°¡ ¼³Á¤µÇ¾î ÀÖÀ¸¸é $php_errormsgÀÌ¶ó´Â Àü¿ª °Ë¼ö¿¡ ¸¶Áö¸·À¸·Î ¹ß»ýÇÑ ¿¡·¯ ¸Þ½ÃÁö°¡ µé¾îÀÖ°Ô µÈ´Ù.

track_vars boolean
ÀÌ ÁöÌÃÀÚ°¡ ¼³Á¤µÇ¸é, °¢°¢ HTTP_GET_VARS, HTTP_POST_VARS, HTTP_COOKIE_VARSÀÇ Àü¿ª ¹è¿¿¡ GET / POST / cookie °Ë¼öµéÀÇ ÀÔ·ÂÀ» ÀúÀåÇÏ°Ô µÈ´Ù.

upload_tmp_dir string
ÆÄÀÏ ¾÷·Îµå½Ã ÆÄÀÌÀ» ÀúÀåÇÒ ÀÓ½Ã µð·º°Ää¸®¸¦ ÁöÁ¤ÇÑ´Ù. PHP°¡ µ¿ÀûÇÏ°ÔµÇ´Â User ID¿¡ ´ëÇØ ¾²±â ±ÇÑÑÀÌ ÀÖ¾î¾ß ÇÑ´Ù.

user_dir string
PHP ÆÄÀÏµéÀ» À§ÇÑ »ç¿ëÀÚÀÇ È¨ µð·º°Ää¸®ÀÇ base ÀÌ¸§À» ÁöÁ¤ÇÑ´Ù. (Ex. public_html)

warn_plus_overloading boolean
ÀÌ°ÍÀÌ ¼³Á¤µÇ¸é, ¹®ÀÚ¿-¿¡ ´õÇÏ±â(+) ¿¬»êÀÚ°¡ »ç¿ëµÉ ¶§ PHP°¡ ÁÖÀÇ ¸Þ½ÃÁö¸¦ Ãâ·ÂÇÏ°Ô ¸¸µç´Ù. ÀÌ ¿É¼ÇÀº ½º¸®½º¸®ÀÇ ´õÇÏ±â ¿¬»êÀÚ¸¦ ¹®ÀÚ¿- concatenator(.)·Î ´Ù¸¥ ÀÛ¾ºÇÏ´Âµ¥ µµ¿òÀ» ÁØ´Ù.

Mail Configuration Directives

SMTP string
Windows¨°æ¿¡¼- PHP°¡ »ç¿ëÇÒ SMTP ¼-¹öÀÇ DNS ÀÌ¸§ÀÌ³ª IP ¹øÈ£. **mail()** ÇÔ¼ö·Î ÆíÁö¸¦ °¡³»·Á¸é ÀÌ ¿É¼ÇÀ» ¹Ýµå½Ã ¼³Á¤ÇØ ÁÖ¾î¾ß ÇÑ´Ù.

sendmail_from string
Windows¨°æ¿¡¼- PHP°¡ ÆíÁö¸¦ °¡³¾ ¶§ "From:"¿¡ »ç¿ëµÉ ¸ÞÀÏ ¼îµå·¹½º.

sendmail_path string
sendmail ÇÁ·Î±×·¥À» Ã£À» Path. °¡ÅëÀº /usr/sbin/sendmail ÀÌ³ª /usr/lib/sendmail ¿¡ ÀÖ´Ù. **configure** ÇÁ·Î±×·¥ÀÌ sendmailÀÇ À§Ä¡¸¦ Ã£¾Æ ÀÌ°ÍÀ» ±â°»°ªÀ¸·Î ¸¸µçÁö ¾Ê¾Ò°Å³ª ¿À·ù¸¦ ÀÖ¾» ¶§, ¿©¨°ÐÀº ¿©±â¼-Á÷Á¢ ÁöÁ¤ÇØ ÁÙ ¼ö ÀÖ´Ù.

sendmailÀ» »ç¿ëÇÏÁö ¾Ê´Â ½Ã½º¹ÛÀÇ °æ¿ì ÇöÀç Á¦°øµÇ°í ÀÖ´Â ´Ù¸¥ ÞÀÏ ½Ã½º¹ÛÀÇ sendmail È£È¨ ¸í·É(wrapper/replacement)À¸·Î ¼³Á¤ÇØ ÁÖ¾î¾ß ÇÑ´Ù. ¿¹¸¦µé¾î Qmail »ç¿ëÀÚ´Â °¡Åë /var/qmail/bin/sendmail·Î ¼³Á¤ÇØ ÁÖ¸é µÈ´Ù.

Safe Mode Configuration Directives

safe_mode boolean
PHP¸¦ ¾ÈÀüÇü ¸ðµå(safe mode)·Î ÀÛµµÇÅ ÇÑ´Ù.

safe_mode_exec_dir string
PHP°¡ ¾ÈÀüÇü ¸ðµå¿¡¼- ÀÛµµÇÒ ¶§, **system()** ÀÌ³ª ±âÅ¸ ´Ù¸¥ ÇÁ·Î±×·¥À» ½ÇÇà ½ÃÅ°´Â ÇÔ¼öµéÀº ÀÌ µð·º°Ää¸®¿¡ ÀÖ´Â ÇÁ·Î±×·¥À» ¾Æ´Ï¸é ÀÛ¾÷À» °Å°ÎÇÑ´Ù.

Debugger Configuration Directives

debugger.host string
µð¹ö°Å°¡ »ç¿ëÇÒ hostÀÇ DNS name ÀÌ³ª IP address

debugger.port string

µð¹ö°Å°¡ »ç¿ëÇÒ Port ¹øÈ£

debugger.enabled boolean
µð¹ö°Å¸¦ ÀÛµ¿½ÃÅ²´Ù.

Extension Loading Directives

enable_dl boolean
ÀÌ Áö½ÃÀÚ´Â PHP¸¦ ¾ÆÆÄÄ¡ ¸ðµâ·Î ½ÇÇàÇÒ ¶§¸¸ È¿·ÂÀ» °¡Áø´Ù. ¿©·¯°ÐÀº PHP¿¡¼- **dl()** À» »ç¿ëÇÏ¿© extensionÀ» dynamic loadingÇÏ´Â ±â´ÉÀ» virtual server³ª µð·ºÄä¸® °°·Î °¡ÉÇÏ°Ô ÇÏ°Å³ª ºÒ¡´ÉÇÏ°Ô ÇÒ ¼ö ÀÖ´Ù.

dynamic loadingÀ» ºÒ°¡´ÉÇÏ°Ô ÇÏ´Â Ã¹ ¹ø Â° ÀÌÀ¯´Â ¸¸¿È»óÀÇ ÀÌÀ¯ÀÌ´Ù. dynamic loading¿¡¼-´Â ¸ðµç safe_mode ¼³Á¤°ú open_basedir ¼³Á¤ÀÌ ¹«½Ãµé ¼ö ÀÖ±â ¶§¹®ÀÌ´Ù.

±â°»°ªÀº ¸ðµç dynamic loadingÀ» Çã¿ëÇÏ´Â °ÍÀÌ´Ù. ´Ü, safe-mode¸¦ »ç¿ëÇÏ´Â Áß¿¡´Â Ç×»ó **dl()**ÀÇ »ç¿ëÀÌ ºÒ°¡´É ÇØ Á¤´Ù.

extension_dir string
µ¿ÀûÀ¸·Î ºÒ·¯¿Ã(dynamically loadable) extensionÀ» Ã£À» µð·ºÄä¸®¸¦ ¼³Á¤ÇÑ´Ù.

extension string
PHP°¡ ½ÃÀÛµÉ ¶§ ·ÎµåÇÒ dynamically loadable extensionµéÀ» ¼³Á¤ÇÑ´Ù.

MySQL Configuration Directives

mysql.allow_persistent boolean
¿µ±¸Àû·Î(persistent) MySQLÀ» Á¢¼ÓÇÒ ¼ö ÀÖ°Ô ÇÑ´Ù.

mysql.max_persistent integer
ÇÁ·Î¼¼½º´ç ¿µ±¸ÀûÀÎ(persistent) MySQL Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

mysql.max_links integer
¿µ±¸ÀûÀÎ(persistent) Á¢¼Ó» Æ÷ÇÔÇÑ ÇÁ·Î¼¼½º´ç MySQL Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

---

mSQL Configuration Directives

msql.allow_persistent boolean
¿µ±¸ÀûÀ¸·Î(persistent) mSQLÀ» Á¢¼ÓÇÒ ¼ö ÀÖ°Ô ÇÑ´Ù.

msql.max_persistent integer
ÇÁ·Î¼¼½º´ç ¿µ±¸ÀûÀÎ(persistent) mSQL Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

msql.max_links integer
¿µ±¸ÀûÀÎ(persistent) Á¢¼Ó» Æ÷ÇÔÇÑ ÇÁ·Î¼¼½º´ç mSQL Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

Postgres Configuration Directives

pgsql.allow_persistent boolean
¿µ±¸ÀûÀ¸·Î(persistent) Postgres¸¦ Á¢¼ÓÇÒ ¼ö ÀÖ°Ô ÇÑ´Ù.

pgsql.max_persistent integer
ÇÁ·Î¼¼½º´ç ¿µ±¸ÀûÀÎ(persistent) Postgres Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

pgsql.max_links integer
¿µ±¸ÀûÀÎ(persistent) Á¢¼Ó» Æ÷ÇÔÇÑ ÇÁ·Î¼¼½º´ç Postgres Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

Sybase Configuration Directives

sybase.allow_persistent boolean
¿µ±¸ÀûÀ¸·Î(persistent) Sybase¸¦ Á¢¼ÓÇÒ ¼ö ÀÖ°Ô ÇÑ´Ù.

sybase.max_persistent integer
ÇÁ·Î¼¼½º´ç ¿µ±¸ÀûÀÎ(persistent) Sybase Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

sybase.max_links integer
¿µ±¸ÀûÀÎ(persistent) Á¢¼Ó» Æ÷ÇÔÇÑ ÇÁ·Î¼¼½º´ç Sybase Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

Sybase - CT Configuration Directives

sybct.allow_persistent boolean
¿µ±¸ÀûÀ¸·Î(persistent) Sybase - CT¸¦ Á¢¼ÓÇÒ ¼ö ÀÖ°Ô ÇÑ´Ù.

sybct.max_persistent integer
ÇÁ·Î¼¼½º´ç ¿µ±¸ÀûÀÎ(persistent) Sybase - CT Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

sybct.max_links integer
¿µ±¸ÀûÀÎ(persistent) Á¢¼ÓÀ» Æ÷ÇÔÇÑ ÇÁ·Î¼¼½º´ç Sybase - CT Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

sybct.min_server_severity integer
severity¸¦ °¡Áö´Â ¼-¹ö ¸Þ½ÃÁö °³¼ö°¡ sybct.min_server_severity¿¡ ¼³Á¤ÇÑ °ª¸¸ Å©°Å³ª °Ø¾ÆÁö¸é warningÀ» ·¹Æ÷Æ®ÇÑ´Ù. ÀÌ °ªÀº ½ºÅ©¸³Æ®¿¡¼- **sybase_min_server_severity()**ÇÔ¼ö¸¦ ÅëÇØ ¼³Á¤ÇÒ ¼öµµ ÀÖ´Ù. ±â» °ªÀº 10ÀÌ´Ù.

sybct.min_client_severity integer
severity¸¦ °¡Áö´Â Å¬¶óÀÌ¾ðÆ® ¶óÀ̰ê·¯¸® ¸Þ½ÃÁö °³¼ö°¡ sybct.min_server_severity¿¡ ¼³Á¤ÇÑ °ª¸¸ Å©°Å³ª °Ø¾Æ Áö¸é warningÀ» ·¹Æ÷Æ®ÇÑ´Ù. ÀÌ °ªÀº ½ºÅ©¸³Æ®¿¡¼- **sybase_min_client_severity()**ÇÔ¼ö¸¦ ÅëÇØ ¼³Á¤ÇÒ ¼öµµ ÀÖ´Ù. ±â» °ªÀº 10ÀÌ´Ù.

sybct.login_timeout integer
¼-¹ö¿¡ ¿¬°áÀ» ½ÃµµÇÏ°í ±â´Ù¸®´Â ÃÖ´ë ½Ã°£. ´ÜÀ§´Â ÃÊÀÌ°í ÁöÁ¤µÈ ½Ã°£¿¡ ¿¬°áÀÌ ¿Ï·áµÇÁö ¸øÇÏ¸é ¿¬°á µ¿ÀÛÀº ½Ç ÆÐÇÏ´Â °ÍÀÌ´Ù. ¸¸¾à ¿¬°á½Ãµµ ÁßmaxexecutiontimeÀÇ ¼³Á¤Àº ÃÊ°úÇÏ°Ô µÇ¸é ½ºÅ©¸³Æ®´Â ¿¬°á ½ÇÆÐ¸¦ ¾Ë¸®±â Àü¿¡ Á¾·áµÇ¾î ¹ö¸´Ù. ±â»°ªÀº 1ºÐÀÌ´Ù.

sybct.timeout integer
selectdb³ª queryµîÀÇ ¸í·ÉÀ» ÁÖ°í ÀÀ´äÀ» ±â´Ù¸®´Â ÃÖ´ë ½Ã°£. ´ÜÀ§´Â ÃÊÀÌ°í ÁöÁ¤µÈ ½Ã°£¾È¿¡ ÀÀ´äÀÌ ¾øÀ¸¸é ÀÌ µ¿ÀÛÀº ½ÇÆÐÇÏ´Â °ÍÀÌ´Ù. ¸¸¾à ¿¬°á½Ãµµ ÁßmaxexecutiontimeÀÇ ¼³Á¤Àº ÃÊ°úÇÏ°Ô µÇ¸é ½ºÅ©¸³Æ®´Â µ¿ÀÛ ½ÇÆÐ¸¦ ¾Ë¸®±â Àü¿¡ Á¾·áµÇ¾î ¹ö¸´Ù. ±â»°ªÀº ¾ø°í, ¹«ÇÑ¤ ±â´Ù¸®´Â °ÍÀÌ´Ù.

sybct.hostname string
sp_who¿¡ ÀÇÇØ Ç¥½ÃµÉ ¿¬°áÀ» ¿äÃ»ÇÏ´Â hostÀÇ ÀÌ¸§. ±â»°ªÀº ¾ø´Ù.

BC Math Configuration Directives

bcmath.scale integer
bc ¼öÇÐ ÇÔ¼öµéÀÌ »ç¿ëÇÏ´Â ¼Ò¼öÁ¡ÀÌÇÏ ÀÚ¸®¼ö.

Browser Capability Configuration Directives

browscap string
browser ±â´É(capabilities) ÆÄÀÏÀÇ ÀÌ¸§

Unified ODBC Configuration Directives

uodbc.default_db string
odbc_connect()³ª **odbc_pconnect()**¿¡¼- noneÀ¸·Î ¼³Á¤µÇ¾î ÀÖÀ» ¶§ »ç¿ëÇÒ ODBC Data Source¸¦ ÁöÁ¤ÇÑ´Ù.

uodbc.default_user string
odbc_connect()³ª **odbc_pconnect()**¿¡¼- noneÀ¸·Î ¼³Á¤µÇ¾î ÀÖÀ» ¶§ »ç¿ëÇÒ User ÀÌ¸§À» ÁöÁ¤ÇÑ´Ù.

uodbc.default_pw string
odbc_connect()³ª **odbc_pconnect()**¿¡¼- noneÀ¸·Î ¼³Á¤µÇ¾î ÀÖÀ» ¶§ »ç¿ëÇÒ Password¸¦ ÁöÁ¤ÇÑ´Ù.

uodbc.allow_persistent boolean
¿µ±¸ÀûÀ¸·Î(persistent) ODBC¸¦ Á¢¼ÓÇÒ ¼ö ÀÖ°Ô ÇÑ´Ù.

uodbc.max_persistent integer
ÇÁ·Î¼¼½º´ç ¿µ±¸ÀûÀÎ(persistent) ODBC Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

uodbc.max_links integer
¿µ±¸ÀûÀÎ(persistent) Á¢¼ÓÀ» Æ÷ÇÔÇÑ ÇÁ·Î¼¼½º´ç ODBC Á¢¼ÓÀÇ ÃÖ´ë °³¼ö

Apache Module

Apache module configuration directives

CGI redirection module/action module

CGI

Virtual hosts

Security

PHP´Â °·ÂÇÑ ¾ð¾îÀÌµð. ±× ÀÎÅÍÇÁ¸®ÅÍ´Â À¥¼¹öÀÇ ¸ðµâ·Î µÇ¾î ÀÖ°Ç CGI binary·Î µÇ¾î ÀÖ°Ç °£¿¡, ÆÄÀÏÀ» »ç¿ëÇÒ ¼ö ÀÖ°í, ´Ù¸¥ ¸í·ÉÀÌ³ª ÇÁ·Î±×·¥À» ½ÇÇàÇÒ ¼ö ÀÖ°í, ¼¹ö·Î ³×Æ®¿÷ ¿¬°áÀ» ÇÒ ¼öµµ ÀÖ´Ù. ÀÌ·± Æ¯¼ºµéÀº ±âº»ÀûÀ¸·Î ¾î¶² ÇÁ·Î±×·¥ÀÌµç ½ÇÇà½ÃÄÑ À¥¼¹ö¸¦ °ÔÀÌÆÃÇÒ ¼ö ÀÖ´Ù. PHP´Â °Ð¸íÈ÷ PerlÀÌ³ª C°¸´Ù Á»´õ ¾ÈÀüÇÑ CGI ÇÁ·Î±×·¥À» ¸¸µé ¼ö ÀÖ´Ù. ÄÄÆÄÀÏ½Ã³ª ½ÇÇà½Ã ¼³Á¤ ¿É¼ÇµéÀ» ÀûÀýÈ÷ ¼±ÅÃÇÔÀ¸·Î, ¿©·¯ºÐÀº ¿øÇÏ´Â Á¤µµÀÇ ÀÚÀ¯·Î¿ò°ú ¸¸È»óÀÇ Á¦¾àÀ» Á¶È½ÃÅ³ ¼ö ÀÖ´Ù.

PHP´Â ¼ö¸¹Àº ¹æ¹ýÀ¸·Î ÀÌ¿ëÀÌ °¡´ÉÇÏ°í, ÀÌ¿¡ µû¸¥ ¿©·¯ °¡Áö ¼³Á¤ ¿É¼ÇÀÌ ÀÖÀ» ¼ö ÀÖ´Ù. ¸¹Àº ¿É¼Ç ¼±ÅÃ »ç×µéÀº PHP¸¦ °¸´Ù ¸¹Àº ¸ñÀû¸·Î »ç¿ëÇÒ ¼ö ÀÖµµ·Ï ÇÏÁö¸¸, ÀÌ ¿É¼ÇµéÀÇ Á¶ÇÕ°ú ¼¹öÀÇ ¼³Á¤¿¡ µû¶ó °¸È»óÀÇ ÇãÁ¡À» ³ëÃâ½ÃÅ³ ¼öµµ ÀÖ´Ù. ÀÌ ÀåÀº ´Ù¸¥ ¿É¼ÇÀÇ Á¶ÇÕ ¹æ¹ý°ú ÀÌ°ÍÀ» ¾ÈÀüÇÏ°Ô »ç¿ëÇÏ´Â »óÈ²À» ¼³¸íÇÑ´Ù.

CGI binary

°¡´ÉÇÑ °ø°Ý (Possible attacks)

CGI binary ÇüÅÂÀÇ PHP´Â ¾î¶² ÀÌÀ¯·Î ¼¹ö ¼ÒÇÁÆ®¿þ¾î(¾ÆÆÄÄ¡ °°Àº)ÀÇ ¸ðµâ·Î »ç¿ëµÇ±â¸¦ ¿øÇÏÁö ¾Ê´Â °æ¿ì¿¡ »ç¿ëÇÒ ¼ö ÀÖ°í, PHP¸¦ ´Ù¸¥ Á¾·ùÀÇ CGI wrapper¿Í Çô²² ¾ÈÀüÇÑ chroot¿Í setuid È¯°æÀ» ¸¸µé¾î ³»´Â °÷¿¡ ¶§¿¡ »ç¿ëÇÒ ¼ö ÀÖ´Ù. ÀÌ ¼³Á¤À° °¸Åë ½ÇÇàÇÒ PHP binary¸¦ À¥¼¹öÀÇ cgi-binµð·ºÅä¸®¿¡ ¼³Ä¡ÇØ¾ß ÇÑ´Ù. CERT ±Ç°í»çÇ× CA-96.11¿¡ µû¸£¸é ¸ðµç ÀÎÅÍÇÁ¸®ÅÍµéÀº cgi-bin¿¡ ¼³Ä¡ÇÏµµ·Ï ±Ç°íµÈ´Ù. °ñ·Ï PHP binary°¡ µ¶¸³ÀûÀÎ ÀÎÅÍÇÁ¸®ÅÍ·Î »ç¿ëµÈ´Ù ÇÏ´õ¶óµµ, PHP´Â ÀÌ ¼³Á¤ÀÌ ¸¸µé¾î³¾ °¡´ÉÇÑ °ø°ÝÀ» ´ÙÀ½°ú °°ÀÌ ¸·¾Æ³»µµ·Ï ¼³°èµÇ¾î ÀÖ´Ù. :

- ½Ã½ºÅÛ ÆÄÀÏ·ÎÀÇ Á¢±Ù : `http://my.host/cgi-bin/php?/etc/passwd`

  ÀÌ url³»ÀÇ ¹°À½Ç¥(?)ÀÌÈÄ¿¡ ÀÖ´Â ¿äÃ»À° CGI ÀÎÅÍÆäÀÌ½º°¡ ÀÇÇØ command line argument·Î ÀÎÅÍÇÁ¸®ÅÍ¿¡°Ô Àü´ÞµÈ´Ù. °¸Åë ÀÎÅÍÇÁ¸®ÅÍ´Â Ä¿¸Çµå ¶óÀÎÀÇ Ã¹ ¹øÂ° argumentÀÇ ÆÄÀÏÀ» ÀÐ¾î¼ ½ÇÇà ÇÑ´Ù.

  CGI binary·Î ½ÇÇà µÈ PHP´Â ¸ðµç command line argumentµéÀ» ¸ðµÎ ¹«½ÃÇÑ´Ù.

- ¼¹öÀÇ À¥ ¹®¼·ÎÀÇ Á¢±Ù : `http://my.host/cgi-bin/php/secret/doc.html`

  ÀÌ urlÀÇ PHP binary ÀÌÈÄÀÇ path Á¤°¸(`/secret/doc.html`)´Â ÀÏ¹ÝÀûÀ¸·Î CGI ÇÁ·Î±×·¥¿¡ ÀÇÇØ ÀÐ°í Ç¼®µÇ¾î Áö´Â ÆÄÀÏ ÀÌ¸§À¸·Î »ç¿ëµÈ´Ù. °¸Åë ¾î¶² À¥¼¹ö ¼³Á¤ Áö½ÃÀÚ(ApacheÀÇ °æ¿ì Action)°¡ `http://my.host/secret/script.php3`¿Í °°Àº ¹®¼ ¿äÃ»À» PHP ÀÎÅÍÇÁ®·Î ¸®µð·ºÆ® ÇÏ´Âµ¥ »ç¿ëµÈ´Ù. ÀÌ ¸°Ô ¼³Á¤ÇÏ¸é, À¥¼¹ö´Â ¿ì¼± `/secret` µð·ºÅä¸®ÀÇ Á¢±Ù ±ÇÇÑÀ» °Ë»çÇÑ ÈÄ¿¡ `http://my.host/cgi-bin/php/secret/script.php3`ÀÇ ¸®µð·º®® ¿äÃ»À» ÇÏ°Ô µÈ´Ù. °ÒÇàÈ÷µµ, ÀÌ ¿äÃ»ÀÌ ¿ø·¡ÀÇ form¿¡¼ ÁÖ¾îÁø °ÍÀÌ¶ó¸é, À¥¼¹ö´Â `/secret/script.php3` ÆÄÀÏ¿¡ ´ëÇÑ Á¢±Ù ±ÇÇÑ °Ë»ç¸¦ ÇÏÁö ¾Ê°í, `/cgi-bin/php` ÆÄÀÏ¿¡ ´ëÇÑ °Ë»ç¸¸À» ÇàÇÑ´Ù. ÀÌ ¹æ¹ýÀ» »ç¿ëÇÏ¸é `/cgi-bin/php`¿¡ Á¢±Ù °¡´ÉÇÑ »ç¿ëÀÚ´Â À¥¼¹ö»óÀÇ ¸ðµç °¸È£µÈ ¹®¼µé Á¢±ÙÇÒ ¼ö ÀÖ´Ù.

  PHP¿¡¼´Â ¼¹ö ¹®¼ tree°¡ Á¢±Ù Á¦¾àÀÌ ÀÖ´Â µð·ºÅä¸®¸¦ °¡Áö°í ÀÖ´Ù¸é, ÄÄÆÄÀÏ½Ã¿¡ --enable-force-cgi-redirect ¼³Á¤ ¿É¼ÇÀ» ÁÖ°í, ½ÇÇà½Ã¿¡ doc_root¿Í user_dir Áö½ÃÀÚ¸¦ »ç¿ëÇÏ¿©¼ ÀÌ °ø°ÝÀ» ¸·¾Æ³¾ ¼ö ÀÖ´Ù. ¾Æ·¡¿¡ ¿©·¯ °¡Áö ´Ù¸¥ Á¶ÇÕ ¹æ¹ý¿¡ ´ëÇÑ ÀÚ¼¼ÇÑ ¼³¸íÀÌ ³ª¿Í ÀÖ´Ù.

Case 1: °ø¿ëÆÄÀÏ¸¸ Á¦°øÇÏ±â (only public files served)

¸¸¾à ¿©·¯ºÐÀÇ ¼¹ö°¡ password ³ª ip ±âÁØÀÇ Á¢±Ù ÅëÁ¦¿¡ °üÇÑ ¾î¶°ÇÑ »çÇ×µµ ¾ø´Ù¸é, ÀÌ·¯ÇÑ ¼³Á¤ ¿É¼ÇµéÀº ÇÊ¿ä ¾ø´Ù. ¸¸¾à ¿©·¯ºÐÀÇ À¥¼¹ö°¡ ¸®µð·ºÆ®¸¦ Çã¿ëÇÏÁö ¾Ê°Å³ª, ¼¹ö°¡ ÇØ´ç ¿ä±¸°¡ ¾ÈÀüÇÑ ¸®µð·ºÆ®µÈ ¿ä±¸ÀÎÁö ´ëÇØ PHP binary¿Í Åë½ÅÇÒ ¼ö ÀÖ´Â ¹æ¹ýÀÌ ¾øÀ» °æ¿ì, ¿©·¯ºÐÀº ½°Å©¸³Æ®¸¦ ¼³Á¤ÇÏ±â À§ÇØ --disable-force-cgi-redirect ¿É¼ÇÀ» ¸í½ÃÇÒ ¼ö ÀÖ´Ù. ¿©·¯ºÐÀº ¿©ÀüÈ÷ ¿©·¯ºÐÀÇ PHP ½°Å©¸³Æ°¡, `http://my.host/cgi-bin/php/dir/script.php3`À³·³ Á÷Á¢ Á¢±ÙÇÏ´Â °Íµµ ¾Æ´Ï°í `http://my.host/dir/script.php3`°°ÀÌ ¸®µð·º¼Ç¿¡ ÀÇÇØ¼µµ ¾Æ´Ñ, ´Ù¥ ¹ÏÁö ¸øÇÒ ¹æ¹ý¿¡ ÀÇÇØ º¸·Á Áú ¶§¸¦ ´ëºñ ÇÏ¿© ´ëÃ¥À» °±¸ÇØ µÎ¾î¾ß ÇÕ´Ï´Ù.

¸®µð·º¼ÇÀº ¿¹¸¦µé¾î Apache¿¡¼´Â AddHandle°ú Action Áö½ÃÀÚ¸¦ »ç¿ëÇÏ¿© ¼³Á¤ÇÒ ¼ö ÀÖ½À´Ï´Ù. (¾Æ·¡¸¦ °¸¼¼¿ä.)

Case 2: --enable-force-cgi-redirect ¿É¼Ç »ç¿ë (using --enable-force-cgi-redirect)

ÀÌ ÄÄÆÄÀÏ½ÃÀÇ ¿É¼ÇÀº »ç¿ëÀÚ°¡ `http://my.host/cgi-bin/php/secretdir/script.php3`¿Í °°ÀÌ PHP¸¦ Á÷Á¢ È£ÃâÇÏ´Â °ÍÀ» ¸·¾ÆÁØ´Ù. ´ë½Å, ÀÌ ¸ðµå¿¡¼ PHP´Â ±×°ÍÀÌ À¥¼¹öÀÇ ¸®µð·ºÆ® ±ÔÄ¢À» Åë°úÇÑ °æ¿ì¿¡¸¸ parse ÀÛ¾÷À» ÇÒ °ÍÀÌ´Ù.

ÀÌ¹ÝÀûÀ¸·Î ¾ÆÆÄÄ¡ ¼³Á¤¿¡¼ ¸®µð·ºÆ® ¼³Á¤Àº ´ÙÀ½°ú °°ÀÌ Áö½ÃÀÚ¸¦ »ç¿ëÇÏ¿© ÇÑ´Ù. :

```
Action php3-script /cgi-bin/php
AddHandler php3-script .php3
```

ÀÌ ¿É¼ÇÀº ¾ÆÆÄÄ¡ À¥¼¹ö¿¡¼ ½ÃÇèÀ» °ÅÄ£°í, ¾ÆÆÄÄ¡´Â ¸®µð·ºÆ®µÈ ¿äÃ»¿¡ ´ëÇØ¼´Â REDIRECT_STATUS ¶ó´Â °ñ Ç¥ÁØ CGI È¯°æ °¯¼ö¸¦ ¼³Á¤ÇÑ´Ù. ¸¸¾à ´ç½ÅÀÇ À¥¼¹ö°¡ ¹®¼ ¿äÃ»°¡ Á÷Á¢(direct)ÀÎÁö ¸®µð·ºÆ®(redirect)ÀÎÁö ±¸°ÇÒ ¼ö ÀÖ´Â ¹æ¹ýÀ» Á¦°øÇÏÁö ¾Ê´Â´Ù¸é ¿©·¯ºÐÀº ÀÌ ¿É¼ÇÀ» »ç¿ëÇÒ ¼ö ¾ø´Ù. µû¶ó¼, ¿©·¯ºÐÀº ÀÌ ¹®¼¿¡ ¾º¿©Á® ÀÖ´Â CGI ¹öÀüÀÇ ´Ù¸¥ ½ÇÇà ¹æ¹ýÀ» »ç¿ëÇØ¾ß ÇÒ °ÍÀÌ´Ù.

Case 3: doc_root³ª user_dirÀÇ ¼³Á¤ (setting doc_root or user_dir)

À¥¼¹ö ¹®¼ µð·ºÅä¸®¿¡ script³ª ½ÇÇàÆÄÀÏ °°Àº Active content¸¦ Æ÷ÇÔÇÏ´Â °ÍÀº Á¾Á¾ º¸¾ÈÇÏ´Ù°í »ý°¢µÇ¾î Áø´Ù. ¸¸¾à ¾à°£ÀÇ ¼³Á¤ ¿À·ù·Î ½ºÅ©¸³Æ®°¡ ½ÇÇàµÇÁö ¾Ê°í HTML ¹®¼Ã³·³ ±×´ë·Î Ãâ·Â µÈ´Ù¸é, °¢Á¾ ÁÖÀÇ Àç»êÀÌ³ª Password°°Àº °¸¾È»óÀÇ Á¤º¸°¡ ´©¼³µÇ´Â °á°ú°¡ ÃÊ·¡µÉ Áöµµ ¸ð¸¥´Ù. ±×·¯¹Ç·Î ¸¹Àº ½Ã½ºÅÛ ºü¸®ÀÚµéÀº ½ºÅ©¸³Æ®¸¦ À§ÇÑ °°µµÀÇ µð·ºÅä¸®¸¦ ¸¸µé¾î ³õ°í, ÀÌ µð·ºÅä¸®´Â PHP CGI¸¦ ÅëÇØ¼¸¸ Á¢±ÙÀÌ °¡´ÉÇÏµµ·Ï ¸¸µé¾î ³õ¾Æ ½ÇÇàÀº °¡´ÉÇÏÁö¸¸ º¸¿©ÁöÁö´Â ¾Êµµ·Ï ÇÑ´Ù.

¶ÇÇÑ, ÀüÀå¿¡¼ ±â¼úÇÑ ¹Ù¿Í °°ÀÌ, ¿ä±¸°¡ ¸®µð·ºÆ®µÈ °ÍÀÌ ¾Æ´Ï¶ó°í È®ÀÎÇØÁÖ´Â ¹æ¹ýÀÌ ¾ø´Ù¸é, À¥ ¹®¼ÀÇ root¿Í´Â ´Ù¥ ½ºÅ©¸³Æ®ÀÇ doc_root¸¦ ¼³Á¤ÇØ ÁÖ´Â °ÍÀÌ ¹Ýµå½Ã ÇÊ¿äÇÏ´Ù.

¿©·¯ºÐÀº php3.iniÀÇ doc_root Áö½ÃÀÚ¸¦ ¼³Á¤ÇÏ°Å³ª, PHP_DOCUMENT_ROOT¶ó´Â È¯°æº¯¼ö °ª¿¡ ¼³Á¤ÇØ ÁÖ¾î, PHP ½ºÅ©¸³Æ® ¹®¼ÀÇ root¸¦ ¼³Á¤ÇÒ ¼ö ÀÖ´Ù. ¸¸¾à ÀÌ°ÍÀÌ ¼³Á¤µÇ¾î ÀÖ´Ù¸é, CGI ¹öÀüÀÇ PHP´Â ¾ðÁ¦³ª ÀÌ *doc_root*¿Í request¿¡ ÀÖ´Â °æ·Î Á¤¸¸¦ °¡Áö°í Á¢±ÙÇÏ·Á´Â ÆÄÀÏÀÇ ÀÌ¸§À» ¸¸µç´Ù. µû¶ó¼ ¿©·¯ºÐÀº ÀÌ µð·ºÅä¸® ¹Û¿¡¼´Â ¾î¶°ÇÑ ½ºÅ©¸³Æ®µµ ½ÇÇàµÉ ¼ö ¾øµµ·Ï ¸¸µé°Ô µÈ´Ù. (¾Æ·¡¿¡ ÀÖ´Â *user_dir* ¸¸Àº ¿¹¿ÜÀÌ´Ù)

ÀÌ°÷¿¡¼ ¶Ç ÇÏ³ª »ç¿ëÇÒ ¼ö ÀÖ´Â ¿É¼ÇÀº user_dirÀÌ´Ù. user_dirÀÌ ¼³Á¤µÇ¾î ÀÖÁö ¾ÊÀ¸¸é Á¢±ÙÀ» °¡´ÉÇÑ ÆÄÀÏÀ» *doc_root*¿¡ ÀÖ´Â °Í »ÓÀÌ´Ù. http://my.host/~user/doc.php3°°Àº url·Î´Â user home directory¿¡ ÀÖ´Â ÆÄÀÏÀ» ¿ ¼ö ¾ø´Ù. ´ÜÁö doc_root¾Æ·¡ÀÇ ~user/doc.php3 ÆÄÀÏÀ» °Î¸¦ »ÓÀÌ´Ù. (¹°·Ð Æ¿µ¥[~]·Î ½ÃÀÛÇÏ´Â "~user"¶ó´Â µð·ºÅä¸®ÀÌ´Ù.)

¸¸¾à user_dirÀÌ ¿¹¸¦ µé¾î public_php·Î ¼³Á¤µÇ¾î ÀÖ´Ù¸é, http://my.host/~user/doc.php3¿Í °°Àº ¿ä±¸´Â userÀÇ home µð·ºÅä¸® ¹ØÀÇ public_php¶ó´Â µð·ºÅä¸® ¹ØÀÇ doc.php3¶ó´Â ÆÄÀÏÀ» È£Ãâ ÇÒ °ÍÀÌ´Ù. ¸¸¾à »ç¿ëÀÚÀÇ homeÀÌ /home/user¶ó¸é, ½ÇÇàµÇ´Â ÆÄÀÏÀº /home/user/public_php/doc.php3ÀÌ µÈ´Ù.

user_dir È®ÀåÀº *doc_root* ¼³Á¤°ú °ü·Ã¾øÀÌ ¹ß»ýÇÏ¹Ç·Î, ¿©·¯ºÐÀº document root¿Í user directory Á¢±ÙÀ» ÅëÁ¦¸¦ °°µû·Î ÇÒ ¼ö ÀÖ´Ù.

Case 4: À¥ Æ®¸® ¹Û¿¡ PHP parser (PHP parser outside of web tree)

¸Å¿ì ½Å·ÚÇÒ¼ö ÀÖ´Â ¿É¼ÇÀº¸·Î PHP parser binary¸¦ À¥ Æ®¸® ¹Û¿¡´Ù µÎ´Â ¹æ¹ýÀÌ´Ù. ¿¹¸¦ µé¾î /usr/local/bin °°Àº °÷¿¡ µÐ´Ù. ÀÌ ¿É¼ÇÀÇ ½ÇÁúÀûÀÎ ´ÜÁ¡Àº, ¸ðµç PHP ÅÂÀ» Æ÷ÇÔÇÏ°í ÀÖ´Â ÆÄÀÏÀÇ Ã¹¹øÂ° ¶óÀÎ¿¡ ´ÙÀ½°ú °ñ½ÃÇÏ°Ô ÀûÁîÁÖ´Â °Í »ÓÀÌ´Ù. :

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

¶ÇÇÑ ÆÄÀÏÀ» ½ÇÇà °¡´ÉÇÏ·Î ¸¸µé¾î ÁÖ¾î¾ß ÇÑ´Ù. Áï, ÀÚ½ÅÀÇ ½ÇÇàÀ» À§ÇØ¼ #! shell-escape ¸ÞÄ«´ÏÁòÀ» »ç¿ëÇÏ´Â PerlÀÌ³ª sh È¤Àº ´Ù¸¥ ½ºÅ©¸³Æ® ¾ð¾î¿Í °°Àº ¹æ½ÄÀ¸·Î ´Ù·ç´Â °ÍÀÌ´Ù.

ÀÌ ¼³Á¤¿¡¼ PHP°¡ PATH_INFO¿Í PATH_TRANSLATEDÁ¤º¸¸¦ ¿Ã¹Ù¸£°Ô ´Ù·ç·Á¸é, php parser°¡ --enable-discard-path ¼³Á¤ ¿É¼ÇÀ¸·Î ÄÄÆÄÀÏ µÇ¾î¾ß ÇÑ´Ù.

Apache module

PHP°¡ ¾ÆÆÄÄ¡ÀÇ ¸ðµâ·Î »ç¿ëµÉ ¶§´Â ¾ÆÆÄÄ¡ÀÇ »ç¿ëÀÚ permissionÀ» °è½Â¹Þ´Â´Ù. (ÀÏ¹ÝÀûÀ¸·Î "nobody"·Î ¼³Á¤µÇ¾î ÀÖ´Ù.)

Chapter 5. ±¸¹®°ú ¹®¹ý (Syntax and grammar)

PHPÀÇ ±¸¹®µéÀº ±â°»ÀûÀ¸·Î C¿¡¼ °ô·Á¿Ô´Ù. ¶ÇÇÑ ÀÚ¹Ù¿Í Perl¿¡¼µµ ¿µÇâÀ» ¹Þ¾Ò´Ù.

HTML·Î°ÎÅÍ ºüÁ®³ª¿À±â(Escaping from HTML)

HTML·Î°ÎÅÍ ºüÁ®³ª¿Í "PHP code mode"·Î µé¾î°¡´Â ¹æ¹ýÀº ´ÙÀ½ 3°¡Áö°¡ ÀÖ´Ù :

Example 5-1. Ways of escaping from HTML

```
1.  <? echo("this is the simplest, an SGML processing instruction\n"); ?>
2.  <?php echo("if you want to serve XML documents, do like this\n"); ?>
3.  <script language="php">
```

```
      echo("some editors (like FrontPage) don't like processing instructions");
    </script>
4.  <% echo("As of PHP 3.0.4 you may optionally use ASP-style tags"); %>
```

¸í·É ±¸Ð (Instruction separation)

°¢°¢ÀÇ ¸í·ÉÀº C³ª Perl°ú ¸¶Âù°¡Áö·Î ±¸ÐµÈ´Ù. °¢ ¹®Àå¸¶´Ù ;(¼¼¹ÌÄÝ·Ð)À¸·Î Á¾·áµÈ´Ù.

PHPÀÇ Á¾·áÅÃÀÎ ?>´Â ¹®Àå³¡À¶ó´Â ÀÇ¹Ìµµ ÇÔ²² °¡Áö°í ÀÖ´Ù. µû¶ó¼- ´ÙÀ½ µÎ °³ÀÇ ¹®Àå¸¦ µ¿ÀÏÇÏ´Ù.

```
<?php
    echo("this is example\n");
?>

<? echo("this is example\n") ?>
```

º¯¼ö ÇüÅÂ (Variable types)

PHP´Â ´ÙÀ½ ÇüÅÂÀÇ º¯¼ö typeÀ» Áö¿øÇÑ´Ù. :

PHP supports the following types:

- integer
- double
- string
- array
- object
- pdfdoc (only if enabled PDF support)
- pdfinfo (only if enabled PDF support)

(¿ªÀÚÁÖ : ³»°ÎÀûÀ¸·Î´Â ÇÔ¼ö¿Í Å¬·¡½º¸¸ º¯¼ö·Î ÁöÁ¤ÇÒ ¼öµµ ÀÖÁö¸¸ ÀÌ°ÍéÀº ÀÏ¹ÝÀûÀÎ º¯¼ö·Î Ã¼Å©µÇÁö ¾Ê´Â´Ù.)

º¯¼öÅ¸ÀÔº °¸Ëë ÇÁ·Î±×·¡¸Ó¡ ¼±ÅÃÇÒ ÇÊ¿ä°¡ ¾ø´Ù. ´ë½Å, PHP°¡ ½ÇÇàÁß¿¡ º¯¼öÀÇ °ªÀÇ ³»¿ë¿¡ µû¶ó ÀÚµ¿ÀûÀ¸·Î ¹Ù²Ù¾î ÁØ´Ù.

¸¸¾à ¿©·¯ºÐÀÌ Æ¯Á¤ÇÑ º¯¼ö Å¸ÀÔÀ¸·Î ÁöÁ¤ÇÏ°í ½Í´Ù¸é º¯¼ö¦ cast ÇÏ°Å³ª, **settype()** ÇÔ¼ö¦ »ç¿ëÇÑ´Ù.

Á¾ÁÁ¾ º¯¼öµéÀº ½ÇÇà½Ã ¾î¶² Å¸ÀÔÀ¸·Î µÇ¾î ÀÖ´À³Ä¿¡ µû¶ó Æ¯Á¤ÇÑ »óÈ²¿¡¼- ¿øÇÏ´Â ´ë·Î µ¿ÀÛÇÏÁö ¾ÊÀ» ¶§°¡ ÀÖ´Ù. Type Juggling ÆíÀ» °¸¸é ÀÚ¼¼È÷ ¾Ë ¼ö ÀÖ´Ù.

º¯¼ö ÃÊ±âÈ- (Variable initialization)

PHP¿¡¼- º¯¼ö¦ ÃÊ±âÈ-ÇÏ´Â °ÍÀº ´Ü¼øÈ÷ ÀÌ º¯¼ö¿¡ ¾î¶² °ªÀ» ÁöÁ¤ÇØ ÁÖ¸é µÈ´Ù. Array³ª ObjectµîÀÇ °æ¿ì´Â ³»°ÎÀû À¸·Î Á¶±Ý ´Ù¸¦ ¼öµµ ÀÖÀ¸³ª ´ë°Î°ÐÀÇ °æ¿ì ÀÌ°ÍÀÌ °¡Àå Á÷Á¢ÀûÀÌ°í ¼ö¿ùÇÑ ¹æ¹ýÀÌ´Ù.

¹è¿- ÃÊ±âÈ- (Initializing Arrays)

¹è¿-Àº ÁÖ°¡Áö¹æ¹ýÀ¸·Î ÃÊ±âÈ- µÉ ¼ö ÀÖ´Ù : °ªÀ» ¼øÂ÷ÀûÀ¸·Î ´ëÀÔÇÏ´Â ¹æ¹ý(sequential assign)°ú, **array()**¶ó´Â construct¸¦ »ç¿ëÇÒ ¼ö ÀÖ´Ù. (Array functions °Î°Ð¿¡ ÀÚ¼¼ÇÑ ¾³¸íÀÌ ÀÖ´Ù).

¹è¿-¿¡ ¼øÂ÷ÀûÀ¸·Î °ªÀ» ´ëÀÔÇÏ´Â ¹æ¹ýÀº ¹è¿-¿¡ Ã·ÀÚ ¾øÀÌ ±×³É ´ëÀÔÇÏ¸é µÈ´Ù. ±×·¯¸é »õ·Î ´ëÀÔÇÑ °ªÀº ±× ¹è¿-ÀÇ Á¦ÀÏ ¸¶Áö¸·¿¡ Ã·°¡µÈ´Ù.

```
$names[] = "Jill";   // $names[0] = "Jill"
$names[] = "Jack";   // $names[1] = "Jack"
```

C³ª Perl°ú ¸¶Âù°¡Áö·Î ¹è¿-ÀÇ Ã³À½Àº 1ÀÌ ¾Æ´Ñ 0À¸·Î ½ÃÀÛÇÑ´Ù.

°´Ã¼ÀÇ ÃÊ±âÈ- (Initializing objects)

°´Ã¼(Object)¸¦ ÃÊ±âÈ-ÇÏ´Â ¹æ¹ýÀº new ¸í·ÉÀ» »ç¿ëÏ¿© object¸¦ º¯¼ö¿¡ ÀÎ½ºÅÏ½º ½ÃÅ°´Â °ÍÀÌ´Ù.

```
class foo {
   function do_foo() {
      echo "Doing foo.";
   }
}
$bar = new foo;
$bar->do_foo();
```

º¯¼ö »ç¿ë ¹üÀ§ (Variable Scope)

º¯¼ö¡ »ç¿ë°¡ ´ÉÇÑ °÷À° ±× º¯¼ö°¡ ¼±¾ðµÈ °÷ ¾È¿¡¼- ¸¸ÀÌ´Ù. PHP º¯¼öÀÇ ´ë°Î°ÐÀ° ÇÑ °³ÀÇ ¹üÀ§(single scope)¸¸À» °¡Áö°í ÀÖ´Ù. ±â°»ÀûÀ¸·Î ÇÔ¼ö ¾È¿¡¼- ¼±¾ðµÈ °ÍÀ° ÇÔ¼ö ¾È¿¡¼-¸¸ »ç¿ëµÇ´Â Áö¿ª º¯¼öÀÌ´Ù. ¶ÇÇÑ ÇÔ¼ö ¾È¿¡¼- »ç¿ëµÇ´Â ÇÔ¼öµéÀº ±â°»ÀûÀ¸·Î Áö¿ª º¯¼ö¶ó°í °¡Á¤ÇÏ°í Ã£Ô µÈ´Ù. ¿¹¸¦ µé¾î:

```
$a=1; /* global scope */
Function Test() {
    echo $a; /* reference to local scope variable */
}
Test();
```

ÀÌ ½º¸©¸³Æ®´Â ÇÔ¼ö ¾ÈÀÇ $a º¯¼ö¸¦ Áö¿ªº¯¼ö·Î °¡Á¤ÇÏ¿´°í, Áö¿ªº¯¼ö $a¿¡´Â ¾Æ¹« °ªµµ ´ëÀÔµÇÁö ¾Ê¾Ò±â ¶§¹®¿¡ ¾Æ¹«·± Ãâ·Âµµ ³ªÅ¸³ªÁö ¾Ê´Â´Ù. ÀÌ°ÍÀº C¿Í´Â ¾à°£ Â÷ÀÌ°¡ ³ª´Â ºÎºÐÀÌ´Ù. PHP¿¡¼- Àü¿ªº¯¼ö¸¦ »ç¿ëÇÏ·Á¸é »ç¿ëÇÏ°í ½ÍÀº Àü¿ªº¯¼ö¸¦ ÇÔ¼ö ¾È¿¡ global ¼±¾ðÀ» ÇØ ÁÖ¾î¾ß ÇÑ´Ù. ¿¹¸¦ µé¾î :

```
$a=1;
$b=2;
Function Sum() {
    global $a,$b;
    $b = $a + $b;
}
Sum();
echo $b;
```

À§ÀÇ ½º©¸³Æ®´Â "3"À» Ãâ·ÂÇÒ °ÍÀÌ´Ù. ÇÔ¼ö ³»¿¡¼- $a and $b¸¦ global·Î ¼±¾ðÇßÀ¸¹Ç·Î, ÀÌ ÇÔ¼ö³»ÀÇ $a, $b´Â Àü¿ª º¯¼ö¸¦ ÅüÁ¶ÇÏ°Ô µÈ´Ù. ÇÔ¼ö ³»¿¡¼- ´Ù·ê ¼ö ÀÖ´Â Àü¿ªº¯¼öÀÇ °³¼ö¿¡ ´Â Á¦ÇÑÀÌ ¾ø´Ù.

Àü¿ª º¯¼ö¸¦ ÂüÁ¶ÇÏ´Â µÎ ¹øÂº ¹æ¹ýÀº PHP¿¡¼- Æ¯°°È÷ Á¤ÀÇÇÏ°í ÀÖ´Â $GLOBALS¶ó´Â ¹è¿-À» »ç¿ëÇÏ´Â °ÍÀÌ´Ù. À§ÀÇ ¿¹¸¦ ´ÙÀ½°ú °°ÀÌ ¹Ù²ã ¾µ ¼ö ÀÖ´Ù. :

```
$a=1;
$b=2;
Function Sum() {
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
Sum();
echo $b;
```

$GLOBALS ¹è¿-Àº º¯¼öÀÌ¸§ÀÌ Key°¡ µÇ°í, ±× º¯¼öÀÇ ³»¿ëÀÌ °ªÀÌ µÇ´Â ¿ø¼Ò¸¦ °¡Áø associative ¹è¿-ÀÌ´Ù.

º¯¼ö ¹üÀ§¿¡¼- ¶Ç ´Ù¸¥ Áß¿äÇÑ °ÍÀº Á¤Àû º¯¼ö(static variable)ÀÌ´Ù. Á¤Àû º¯¼ö´Â Áö¿ª ÇÔ¼ö ³»¿¡¼-¸¸ Á¸ÀçÇÏÁö¸¸ Áö¿ª ÇÔ¼ö¿¡¼- Á¾·áµÇ¾îµµ ±× °ªÀ» ÀÐ¾î¹®Áö ¾Ê°í °¡Áö°í ÀÖ´Ù. ´ÙÀ½ ¿¹¸¦ °¸ÀÚ :

```
Function Test() {
    $a=0;
    echo $a;
    $a++;
}
```

À§ÀÇ ÇÔ¼ö´Â Test()¸¦ °Î¸¦ ¶§ ¸¸´Ù $a°¡ 0À¸·Î ´ëÀÔµÇ¹Ç·Î 0À» Ãâ·ÂÇÑ´Ù.

```
Function Test() {
    static $a=0;
    echo $a;
    $a++;
}
```

À§ÀÇ ÇÔ¼ö´Â Test()¸¦ °Î¸¦ ¶§¸¸´Ù Ã³À½¿¡´Â 0, ´ÙÀ½¿¡´Â 1, ±× ´ÙÀ½¿¡´Â 2ÇÏ´Â ½ÄÀ¸·Î ÇÏ³ª¾¿ Áõ°¡µÈ ¼ö¸¦ Ãâ·ÂÇØ ÁØ´Ù. Áï, $aÀÇ °ªÀÌ °¸Á¸µÇ´Â °ÍÀÌ´Ù.

¶ÇÇÑ Á¤Àû º¯¼ö´Â ´ÙÀ½°ú °°ÀÌ recursion¿¡¼- ¹Ýµå½Ã ÇÊ¿äÇÏ´Ù. Àç±ÍÀûÀÎ ÇÔ¼ö´Â ÀÚ±â ÀÚ½ÅÀÇ ÇÔ¼ö¸¦ È£Ãâ¸é Á¤Áö» ¸»´Ù. Àç±ÍÀûÀÎ ÇÔ¼ö¸¦ »ç¿ëÇÒ ¶§´Â ¹«ÇÑ ·çÇÁ¿¡ °üÁú ¼ö ÀÖ±â ¶§¹®¿¡ ¸Å¿ì ÁÖÀÇ¸¦ ¿äÇÑ´Ù. µû¶ó¼- ¿©·¯°ÐÀº Àç±Í È£ÃâÀ» ³¡³»´Â ÀûÀýÇÑ ¹æ¹ýÀ» ¸¶·ÀÇØ µÎ¼îß ÇÑ´Ù. ´ÙÀ½ÀÇ °£´ÜÇÑ ¿¹Á¦´Â 10¹øÀÇ Àç±ÍÀû È£ÃâÀ» ÇÑ´Ù. :

```
Function Test() {
    static $count=0;
    $count++;
    echo $count;
    if($count <  10) {
        Test();
    }
    $count--;
}
```

°¡°¯ °¯¼ö (Variable variables)

°¡²û °¯¼öÀÇ ÀÌ¸§À» °¯°æÇÒ ¼ö ÀÖ´Ù¸é Æí¸®ÇÏ´Ù°í ´À³¥ ¶§°¡ ÀÖ´Ù. ÀÌ°÷¿¡¼´Â °¯¼öÀÇ ÀÌ¸§À» ¹Ù²Ù¾î »ç¿ëÇÏ´Â ¹æ¹ýÀ» ¾Ë¾Æ°¡ÀÚ. °¡ÅëÀÇ °¯¼ö ¼±¾ðÀº ´ÙÀ½°ú °°Ù. :

```
$a = "hello";
```

¿©±â¼ ÀÌ °¡°¯ °¯¼öÀº °¯¼öÀÇ °ªÀ» °¡Áú ¼öµµ ÀÖ°í, ¶ÇÇÑ ÀÌ °ªÀ» °¯¼öÀÇ ÀÌ¸§À¸·Î ´Ù·ê ¼öµµ ÀÖ´Ù. À§ÀÇ ¿¹¿¡¼ *hello*´Â $$¸¦ ¾Õ¿¡ °ÙÀÌ¸é °¯¼ö ÀÌ¸§À¸·Î »ç¿ëÇÒ ¼ö ÀÖ´Â °ÍÀÌ´Ù. ¿¹ :

```
$$a = "world";
```

ÀÌ·¸°Ô ÇÏ¸é PHPÀÇ symbol tree¿¡ ´Â "hello"¶ó´Â °ªÀ» °¡Áö´Â $a¿Í "world"¶ó´Â °ªÀ» °¡Áö´Â $helloÀÇ µÎ °³ÀÇ °¯¼ö°¡ »ý±â°Ô µÈ´Ù. ±×·¯¹Ç·Î ´ÙÀ½°ú °°Àº ¹®ÀåÀº :

```
echo "$a ${$a}";
```

´ÙÀ½ ¹®Àå°ú ¶È °°Àº Ãâ·ÂÀ» ³ªÅ¸³»°Ô µÈ´Ù.:

```
echo "$a $hello";
```

ie. µÑÀÇ Ãâ·Â: *hello world*.

°¡°¯ °¯¼ö¸¦ ¹è¿°ú ÇÔ²² »ç¿ëÇÏ·Á¸é, ÇÑ°¡Áö ¸ðÈ£¼º ¹®Á¦¸¦ ÇØ°áÇØ¾ß ÇÑ´Ù. ±×°ÍÀº, ¸¸¾à ¿©·¯°ÐÀÌ $$a[1]ÀÌ¶ó°í ½èÀ¸é, $a[1]À» ÇÏ³ªÀÇ °¯¼ö·Î º¼ °ÍÀÎÁö, ¾Æ´Ï¸é $$a¸¦ °¯¼ö·Î °¸°¸ ±× °¯¼ö¿¡ [1]ÀÇ ÀÎµ¦½º¸¦ ÇØ ÁÙ °ÍÀÎ°¡ ÇÏ´Â ¹®Á¦ÀÌ´Ù. ´Ù À½°ú °°ÀÌ ÇÏ¿© ¸ðÈ£¼ºÀ» ÇØ°áÇÏÁö. ÀüÀÚÀÇ °æ¿ì ${$a[1]}¶ó°í ¾²°í, ÈÄÀÚÀÇ °æ¿ì´Â ${$a}[1]¶ó°í ¾²¸é µÈ´Ù. (¿ªÀÚÁÖ: °¡°¯ °¯¼ö¸¦ ³ªÅ¸³¾ ¶§´Â Ç×»ó { }·Î µÑ·¯½Î´Â °ÍÀÌ °¸±â ÁÁ°í ¿À·ùµµ ¾ø¾Ù ¼ö ÀÖ´Ù.)

PHP ¹Û¿¡¼ ¿Â °¯¼öµé (Variables from outside PHP)

HTML Forms (GET and POST)

ÆûÀÌ PHP ½ºÅ©¸³Æ®·Î submitµÇ¸é Æû¿¡ ÀÖ´Â ¸ðµç ³»¿ëéÀÌ ÀÚµ¿ÀûÀ¸·Î ¸µé¾îÁø PHP °¯¼ö·Î µé¾î¿Â´Ù. ´ÙÀ½ÀÇ ÆûÀ» °¸ÀÚ :
Example 5 - 2. Simple form variable

```
<form action="foo.php3" method="post">
  Name: <input type="text" name="name"><br>
  <input type="submit">
</form>
```

À§ÀÇ ÆûÀÌ submitµÇ¸é PHP´Â $name°¯¼ö¸¦ ¸¸µé°í, ÀÌ °¯¼ö¿¡ ÆûÀÇ *Name:* ÇÊµå¿¡ ÀÖ·ÂµÈ ¸ðµç ³»¿ëÀ» ÀúÀåÇÑ´Ù.

PHP´Â Æû¿¡ 1Â÷¿ø ¹è¿ °¯¼öµµ »ç¿ëÇÒ ¼ö ÀÖ´Ù. ¿¹¸¦µé¾î, ¿©·¯°ÐÀº ¿©·¯ °¯¼ö¸¦ ÇÔ²² »ç¿ëÇÏ´Â ±×·ì °ü·Ã °¯¼ö(group related variables)³ª multi select °¯¼öÀÇ ³»¿ëéÀ» °Ë»öÇÒ ¼ö ÀÖ´Ù. :
Example 5 - 3. More complex form variables

```
<form action="array.html" method="post">
  Name: <input type="text" name="personal[name]"><br>
  Email: <input type="text" name="personal[email]"><br>
  Beer: <br>
  <select multiple name="beer[]">
    <option value="warthog">Warthog
    <option value="guinness">Guinness
  </select>
  <input type="submit">
</form>
```

¸¸¾à track_vars ¼³Á¤ÀÌ on µÇ¾î ÀÖ°Å³ª, <?php_track_vars?> ÁöÁÄÀÚ°¡ ¼³Á¤µÇ¾î ÀÖ´Ù¸é, POST³ª GETÀ¸·Î ÀüÀûµÇ´Â ¸ðµç °¯¼öµé¸¶´Ù $HTTP_POST_VARS°ú $HTTP_GET_VARS¿¡¼ Ã£À» ¼ö ÀÖ´Ù.

IMAGE SUBMITÀÇ °¯¼ö ÀÌ¸§

ÆûÀº submitÇÒ ¶§ ÀÏ¹ÝÀûÀ¸·Î »ç¿ëÇÏ´Â submit ¹öÆ°´ë¶Å ´ÙÀ½°ú °°ÌÌ ±×¸²À» »ç¿ëÇÒ ¼öµµ ÀÖ´Ù. :

```
<input type=image src="image.gif" name="sub">
```

»ç¿ëÀÚ°¡ image¸¦ Å¬¯ÇÏ¸é, ÆûÀº sub_x¿Í sub_yÀÇ µÎ °³ÀÇ °¯¼ö°¡ Ãß°¡µÇ¾î ¼¹ö¿¡ Àü¼ÛµÈ´Ù. ÀÌ µÎ °¯¼ö´Â ÀÌ¹ÌÁö¿¡¼ »ç¿ëÀÚ°¡ clickÇÑ À§Ä¡ Á¤º¸¸¦ ´ã°í ÀÖ´Ù. ÀÏ°Î °ê¶ó¿ìÀú¿¡¼´Â ¹ØÁÙ(_)´ë½Å ¸¶§Ç¥(.)¸¦ »ç¿ëÇÏ´Â °æ¿ìµµ ÀÖ´Â¥, PHP´Â

ÀÌ·± °æ¿ì ÀÚµ¿ÀûÀ» ¸¶Ä§Ç¥(.)¸¦ ¹ØÁÙ(_)·Î ¹Ù²ãÁØ´Ù.

HTTP Cookies

PHP´Â HTTP ÄíÅ°¸¦ Netscape's SpecÀÇ Á¤ÀÇ¿¡ µû¶ó Á¦°øÇÑ´Ù. Cookie ¸ÞÄ«´ÏÁòÀº Æ®·¡Å·ÀÌ³ª »ç¿ëÀÚ ½Ä°° µîÀ» À§ÇØ ¿ø°Ý browser¿¡ ÀúÀåµÈ µ¥ÀÌÅÍ¸¦ µ¹·Á ¹Þ´Â °úÁ¤À» ¸»ÇÑ´Ù. ¿©·¯°ÐÀº cookie¸¦ ¼³Á¤ÇÏ±â À§ÇØ **SetCookie()** ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼ö ÀÖ´Ù. Cookie´Â HTTP Çì´õÀÇ ÇÑ °Î°ÐÀÌ¹Ç·Î, **SetCookie()** ÇÔ¼ö´Â °ê¶ó¿ìÀú·Î °¸³»´Â ¾î¶² µ¥ÀÌÅÍ°¸´Ùµµ ¾Õ¿¡ »ç¿ëÇØ¾ß ÇÑ´Ù. ÀÌ Á¦¾àÀº **Header()** ÇÔ¼ö¿Í °°Àº Á¦¾àÀ¸·Î °¸¸é µÈ´Ù. ´ç½ÅÀÌ °¸³»ÁØ ¸ðµç cookie´Â ÀÚµ¿ÀûÀ¸·Î GETÀÌ³ª POST ¹æ½Ä µ¥ÀÌÅÍ¿Í °°Àº PHP °¯¼ö·Î °¯È¯µÈ´Ù.

¸¸¾à ¿©·¯°ÐÀÌ µ¿ÀÏÇÑ cookie¿¡ ¿©·¯ °ªÀ» ÀúÀåÇÏ°í ½Í´Ù¸é CookieÀÌ§¿¡ *[]*¸¦ ´õÇÏ¸é µÈ´Ù. ¿¹¸¦ µé¾î :

```
SetCookie("MyCookie[]","Testing", time()+3600);
```

ÄíÅ°´Â »õ·Î ¼³Á¤ÇÏ¸é path³ª µµ¸ÞÀÎÀÌ ´Ù¸£Áö ¾Ê´Â ÇÑ ÀÌÀüÀÇ ÄíÅ°¸¦ µ¤¾î¾²°Ô µÈ´Ù. ±×·¯Ç·Î ¼îÇÎÎ Ä«Æ® °°Àº ÀÀ¿ë ÇÁ·Î ±×·¥¿¡¼´Â Ä«¿îÅÍ¸¦ À¯ÁöÇÏ°í, ÀÌ°ÍÀ» ±âÁØÀ¸·Î Cookie¸¦ »ç¿ëÇÏ´Â °ÍÀÌ ÁÁ´Ù. ¿¹ :
Example 5 - 2. SetCookie Example

```
$Count++;
SetCookie("Count",$Count, time()+3600);
SetCookie("Cart[$Count]",$item, time()+3600);
```

È¯°æ °¯¼ö (Environment variables)

PHP´Â ÀÚµ¿ÀûÀ¸·Î È¯°æ °¯¼öµéÀ» ÀÏ¹ÝÀûÀÎ PHP °¯¼ö·Î ¸¸µç´Ù.

```
echo $HOME;  /* Shows the HOME environment variable, if set. */
```

GET, POST and Cookie ¸ÞÄ«´ÏÁòÀ» ÅëÇØ¼- Á¤º¸¸¦ µé¾î ¿Â Á¤º¸·Î ÀÚµ¿ÀûÀ¸·Î PHP °¯¼ö¡ »ý±â¹Ç·Î, ÀÌ ¹æ¹ýÀº ¶§¶§·Î ¿Ã¹Ù¸¥ ¹öÀüÀ» È®ÀÎÇÏ±â À§ÇØ È¯°æ¸¸·Î °¡´ÉÇÍ °¯¼ö¸¦ ÀÐ¾î µéÀÌ±â¿¡ °¡Å ÁÁ´Ù. ÀÌ¸¦ À§ÇØ **getenv()** ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼öµµ ÀÖ´Ù. ¿©·¯°ÐÀº È¯°æ °¯¼ö¸¦ ¾³Á¤ÇÏ±â À§ÇØ **putenv()** ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼öµµ ÀÖ´Ù.

Server configuration directives

Type ÀüÈ¯ (Type juggling)

PHP´Â °¯¼ö ¼±¾ðº¸´Ù ÀÖ¾î¼- ¸íÈ®ÇÑ Á¤ÀÇ¸¦ ÇÒ ÇÊ¿äµµ ¾ø°í Áö¿øÇÏÁöµµ ¾Ê´Â´Ù. °¯¼öÀÇ Çü½ÅÀº °¯¼ö¡ »ç¿ëµÇ´Â ¹®Åå ¼Ó¿¡¼- °áÁ¤µÈ´Ù. ´Ù¸¥ ¸»·Î ÇÏ¸é, ¸¸¾à ¿©·¯°ÐÀÌ *var*ÀÌ¶ó´Â °¯¼ö¿¡ ¹®ÀÚ¿- °ªÀ» ÇÒ´çÇÏ¸é *var*´Â ¹®ÀÚ¿- °¯¼ö¡ µÇ´Â °ÍÀÌ°í, Á¤¼ö °ªÀ» ÇÒ´çÇÏ¸é Á¤¼ö °¯¼ö¡ µÇ´Â °ÍÀÌ´Ù.

PHPÀÇ ÀÚµ¿ Çü °¯È¯ ¿¹´Â '+' ¿¬»ê¿¡¼- Ã£À» ¼ö ÀÖ´Ù. ¾î¶² ÇÑ ÇÇ ¿¬»êÀÚ°¡ doubleÀÌ¸é ³ª¸ÓÁö ¸ðµç ÇÇ¿¬»êÀÚÀÇ Çüµµ double·Î ¹Ù²î¾î °á°úµµ doubleÇüÀÌ µÈ´Ù. ¸¸¾à ÇÇ¿¬»êÀÚµéÀÌ Á¤¼öÇüÀÌ¸é °á°úµµ Á¤¼öÇüÀÌ´Ù. ¿©±â¼- Áß¿äÇÑ °ÍÀº ÇÇ¿¬»êÀÚ ÀÚ½ÅÀÇ ÇüÀº ¹Ù²îÄö ¾Ê´Â´Ù´Â °ÍÀÌ´Ù.

```
$foo = "0";  // $foo is a string (ASCII 48)
$foo++;      // $foo is the string "1" (ASCII 49)
$foo += 1;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a double (3.3)
$foo = 5 + "10 Little Piggies";    // $foo is a double (15)
$foo = 5 + "10 Small Pigs";   // $foo is an integer (15)
```

¸¶Áö¸· µÎ °³ÀÇ ¿¹°¡ ÀÇ¾ÆÇÑ »ç¶÷À° String conversionÀ» °¸±â ¹Ù¶õ´Ù.

¸¸¾à °¯¼ö¸¦ ¾î¶² Æ¯Á¤Ç typeÀ¸·Î ÄöÁ¤ÇÏ¿© °è»êµÇ°Ô ÇÏ°í ½Í´Ù¸é, Type casting ÆíÀ» °¸±â ¹Ù¶õ´Ù. ¸¸¾à ¿©·¯°ÐÀÌ °¯¼öÀÇ typeÀ» ¹Ù²Ù°í ½Í´Ù¸é **settype()**.À» °¸±â ¹Ù¶õ´Ù.

°¯¼öÀÇ Type ÆÇ´Ü (Determining variable types)

PHP°¡ °¯¼öÀÇ typeÀ» Á¤ÇÏ°í (ÀÏ¹ÝÀûÀ¸·Î) ÇÊ¿äÇÑ °æ¿ì¿¡´Â ±× typeÀ» ¹Ù²Ù¹Ç·Î, Æ¯Á¤ ½ÅÁ¡¿¡ ±× °¯¼ö ¾î¶² typeÀÎÁö´Â ½±°Ô ¾Ë ¼ö ¾ø´Ù. PHP´Â °¯¼öÀÇ typeÀ» ¾Ë¾Æº¸±â ÇÔ¼ö¸¦ ¿©·¯°³ÀÇ ÇÔ¼ö¸¦ °¡Áö°í ÀÖ´Ù. **gettype()**, **is_long()**, **is_double()**, **is_string()**, **is_array()**, and **is_object()**°¡ ±×°ÍµéÀÌ´Ù.

Type casting

PHPÀÇ Type castingÀº C¿¡¼-¿Í °°ÀÌ ¸¸µé°íÀÚ ÀÏÀ» ÇÑ´Ù : °¯È¯ÇÏ·Á´Â °¯¼ö ¾Õ¿¡ ¿øÇÏ´Â type ÀÌ¸§À» °ýÈ£ ¾È¿¡ ½áÁÖ¸é µÈ´Ù.

```
$foo = 10;   // $foo is an integer
$bar = (double) $foo;   // $bar is a double
```

´ÙÀ½°ú °°À° º¯È¯ÀÌ °¡´ÉÇÏ´Ù. :

- (int), (integer) - cast to integer
- (real), (double), (float) - cast to double
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object

ÅÇ°ú °ø¹éÀº °ýÈ£¾È¿¡¼-´Â ¹«ÃµÈ´Ù. Áï ´ÙÀ½ µÎ °³ÀÇ ¹®ÀåÀº µ¿ÀÏÇÏ´Ù. :

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

---

¹®ÀÚ¿- º¯È¯ (String conversion)

¹®ÀÚ¿-ÀÌ ¼ýÀÚ·Î ´Ù·ç¾îÁú ¶§, °á°ú °ª°ú typeÀº ´ÙÀ½°ú °°ÀÌ °áÁ¤µÈ´Ù.

¸¸¾à ¹®ÀÚ¿- ¾È¿¡ '.' ÀÌ³ª 'e', 'E'ÀÇ ¹®ÀÚ°¡ ÀÖÀ» °æ¿ì ±× typeÀº doubleÀÌ µÈ´Ù. ±×·¸Áö ¾Ê´Ù¸é Á¤¼ö °ªÀÌ´Ù

±× °ªÀº ¹®ÀÚ¿-ÀÇ °¡´ÉÑ ºÎºÐ±îÁö¸¸ »ç¿ëµÈ´Ù. ¹®ÀÚ¿-ÀÌ ¿Ã¹Ù¸¥ ¼ýÀÚ µ¥ÀÌÅÍ·Î ½ÃÀÛÇÏÁö ¾ÊÀ¸é ±× °ªÀº 0ÀÌ´Ù. ¿Ã¹Ù¸¥ ¼ýÀÚ µ¥ÀÌÅÍ´Â '+'/'-', '0'-'9','.'°ú ¼ýÀÚ µÚÀÇ 'e'³ª 'E' Ç¥½ÃÀÌ´Ù.

```
$foo = 1 + "10.5";      // $foo is a double (11.5)
$foo = 1 + "-1.3e3";    // $foo is a double (-1299)
$foo = 1 + "bob-1.3e3"; // $foo is a double (1)
$foo = 1 + "bob3";      // $foo is an integer (1)
$foo = 1 + "10 Small Pigs";     // $foo is an integer (11)
$foo = 1 + "10 Little Piggies"; // $foo is a double (11); the string contains 'e'
$foo = "10.0 pigs " + 1;         // $foo is int (11)
$foo = "10.0 pigs " + 1.0;       // $foo is double (11)
```

Ã¹ ¹øÂ° Ç¥Çö½ÄÀÌ stringÀÎ °æ¿ì, º¯¼öÀÇ typeÀº Ã¹ ¹øÂ°° ¼Æ´Ñ µÎ ¹øÂ° Ç¥Çö½Ä¿¡ ÀÇÇØ °áÁ¤µÈ´Ù.

ÀÌ º¯È¯¿¡ ´ëÇÑ ÀÚ¼¼ÇÑ ¼³¸íÀºUnix manual pageÀÇ strtod(3)À» º¸¼¼¶ó.

---

¹®ÀÚ¿- ¿¬°á (String concatination)

PHP´Â µÎ °³ÀÇ ¹®ÀÚ¿-À» ¿¬°á½ÅÅ°´Â ¿¬»êÀÚ·Î Perl°ú °°À° '.'À» »ç¿ëÇÑ´Ù.

```
$oldstring = "abcdef";
$newstring =  "<br>" . $oldstring . "<br>" /* result: "<br>abcdef<br>" */
```

---

¹è¿- (Array manipulation)

ÀÏÂ÷¿ø ¹è¿- (Single Dimension Arrays)

PHP´Â scalar ¹è¿-°ú associative ¹è¿-ÀÇ µÎ °¡Áö¸¦ Áö¿øÇÑ´Ù. »ç½Ç µÎ °¡ÁöÀÇ Â÷ÀÌ´Â ¾ø´Ù. ¿©·¯°ÐÀº **list()**³ª **array()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ¹è¿-À» ¸¸µé°Å³ª, °¢ ¿ø¼ÒÀÇ °ªÀ» Á¤ÇØ ÁÖ¸î ¹è¿-À» ¸¸µé ¼ö ÀÖ´Ù.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

¶ÇÇÑ ´ÙÀ½°ú °°ÀÌ º¯¼ö¿¡ °ªÀ» ´õÇØÁÖ´Â °Í¸¸À¸·Î ¹è¿-À» ¸¸µé ¼öµµ ÀÖ´Ù.

```
$a[] = "hello"; // $a[2] == "hello"

$a[] = "world"; // $a[3] == "world"
```

¹è¿-Àº Á¤·ÄÀ» ¿øÇÏ´Â Çü½Ä¿¡ µû¶ó **asort()**, **arsort()**, **ksort()**, **rsort()**, **sort()**, **uasort()**, **usort()**, **uksort()** ÇÔ¼öµéÀ» ÀÌ¿ëÇØ ¼ø¼-´ë·Î Á¤·ÄÇÒ ¼ö ÀÖ´Ù.

count() ÇÔ¼ö¸¦ »ç¿ëÇÏ¸é ¹è¿-ÀÇ ¿ø¼Ò °³¼ö¸¦ ¼¿ ¼ö ÀÖ´Ù.

next()¿Í **prev()** ÇÔ¼ö¸¦ ÀÌ¿ëÇÏ¿© ¹è¿-ÀÇ ³»¿ëÀ» Å½»öÇÒ ¼ö ÀÖ´Ù. ¹è¿-ÀÇ ³»¿ëÀ» Å½»öÇÏ´Â ¹æ¹ýÀ¸·Î **each()** ÇÔ¼ö¸¦ »ç¿ëÇÒ ¼öµµ ÀÖ´Ù.

´ÙÂ÷¿ø ¹è¿- (Multi - Dimensional Arrays)

´ÙÂ÷¿ø ¹è¿-ÀÌ¶ó ÇÏ´õ¶óµµ ½ÇÁ¦·Î ¸Å¿ì °£´ÜÇÑ´Ù. ¹è¿-ÀÇ °¢ Â÷¿ø¿¡ ´ëÇÏ¿© ´ÜÁö [key]°ªÀ» µÚ¾î ºÙ¿©ÁÖ¸é µÈ´Ù.

```
$a[1]      = $f;              # one dimensional examples
$a["foo"]  = $f;
$a[1][0]     = $f;            # two dimensional
$a["foo"][2] = $f;            # (you can mix numeric and associative indices)
$a[3]["bar"] = $f;            # (you can mix numeric and associative indices)
$a["foo"][4]["bar"][0] = $f;  # four dimensional!
```

¿©·¯ °¡Áö ¹æ¹ýÀ¸·Î ÀÌ ´ÙÂ÷¿ø ¹è¿-¿¡ °ªÀ» ³ÖÀ» ¼ö ÀÖÀ¸³ª, associative ¹è¿-¿¡ °ªÀ» ÀúÀåÇÏ´Â °£´ÜÇÑ ¹æ¹ýÀ¸·Î **array()** ¸í·É Àº »ç¿ëÇÏ´Â ¹æ¹ýÀÌ ÀÖ´Ù. ¿ì¼± ´ÙÀ½ÀÇ µÎ°¡Áö ÄÚµå´Â ÀÏÂ÷¿ø ¹è¿-¿¡ °ªÀ» ÀúÀåÇÏ´Â ¹æ¹ýÀ¸·Î °á°ú´Â µ¿ÀÏÇÏ´Ù.

```
# Example 1:
$a["color"]     = "red";
$a["taste"]     = "sweet";
$a["shape"]     = "round";
$a["name"]      = "apple";
$a[3]           = 4;
# Example 2:
$a = array(
     "color" => "red",
     "taste" => "sweet",
     "shape" => "round",
     "name"  => "apple",
     3       => 4
);
```

array() ÇÔ¼ö´Â ´ÙÂ÷¿ø ¹è¿-¿¡ ´ëÇØ¼-µµ ´ÙÀ½°ú °°ÀÌ »ç¿ëÇÒ ¼ö ÀÖ´Ù. :

```
<?
$a = array(
     "apple"  => array(
          "color"  => "red",
          "taste"  => "sweet",
          "shape"  => "round"
     ),
     "orange"  => array(
          "color"  => "orange",
          "taste"  => "sweet",
          "shape"  => "round"
     ),
     "banana"  => array(
          "color"  => "yellow",
          "taste"  => "paste-y",
          "shape"  => "banana-shaped"
     )
);
echo $a["apple"]["taste"];    # will output "sweet"
?>
```

Chapter 6. ¾ð¾î ±¸¼º (Language constructs)

PHP3 ½º°Å©¸³Æ®´Â ¹®Àå(statement)ÀÇ ³ª¿-·Î ¸¸µé¾îÁø´Ù. ¹®Àå¿¡´Â ´ëÀÔ¹®(assignment), ÇÔ¼öÈ£Ãâ(function call), ¼øÈ¯¹®(loop), Á¶°Ç¹®(conditional statement)µîÀÌ ÀÖ´Ù. ÇÏ³ªÀÇ ¹®ÀåÀº °¸Åë ¼¼¹ÌÄÝ·Ð(;)À¸·Î ³¡³Î´Â´Ù. ¶ÇÑ, ¹®Àåµé°º Áß°ýÈ£({ })¸¦ »ç¿ëÇÏ¿© ÇÏ³ªÀÇ º¹¹®Åå (statement - group)À¸·Î ¸¸µé ¼ö ÀÖ´Ù. ÀÌ º¹¹®ÅåÀº ÇÏ³ªÀÇ ¹®ÀåÃ³·³ »ç¿ëµÉ ¼ö ÀÖ´Ù. ÀÌ Åå¿¡¼-´Â ¿©·¯ °¡Áö ¹®ÅåÀÇ type¿¡ ´ëÇØ ±â¼úµÇ¾î ÀÖ´Ù. (¿ªÀÚÁÖ: ÀÌ°÷ÀÇ ±â°»ÀûÀÎ ¹®¹ýÀº C¸¦ ¾È´Ù°í °¡Á¤ÇÏ°í ¿ø¹®À» ¾à°£ ¿ä¾àÇÏ¿© Á¤¸®ÇÏ¿´À½)

»ó¼ö(Constants)

PHP´Â ¾à°£ÀÇ ±â°»ÀûÀÎ »ó¼öµéÀ» Á¦°øÇÏ°í, ½ÇÇà½Ã »ó¼ö¸¦ ¾³Á¤ÇÒ ¼ö ÀÖ´Â ¸ÞÄ«´ÏÁòÀ» Áö¿øÇÑ´Ù. »ó¼ö´Â °¯¼ö¿¡ºñ ½ÁÇÏ´Ù. ±×·¯³ª **define()**ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ¼±¾ðÇÑ´Ù´Â °Í°ú ³ªÁß¿¡ ´Ù¸¥ °ªÀ¸·Î ¹Ù²Ü ¼ö ¾ø´Ù´Â Á¡ÀÌ Â÷ÀÌ°¡ ÀÖ´Ù.

¹Ì¸® ÁöÁ¤µÈ »ó¼ö´Â ´ÙÀ½°ú °°´Ù.

__FILE__
ÇöÀç Ã³¸®ÁßÀÎ ½º°Å©¸³Æ®ÀÇ ÆÄÀÏ¸í. ÇöÀç ÆÄÀÏÀÌ include³ª require·Î Æ÷ÇÔµÈ ÆÄÀÏÀÌ¶ó¸é, includeÇÔ¼öµîÀ» È£ÃâÇÑ °Î¸ð ÆÄÀÏÀÌ ¾Æ´Ñ, Æ÷ÇÔµÈ ÆÄÀÏÀÇ ÀÌ¸§À» °¡Áø´Ù.

__LINE__
ÇöÀç Ã³¸®ÁßÀÎ ¸í·ÉÀÇ ½º°Å©¸³Æ®³»¿¡¼-ÀÇ ¶óÀÎ ¹øÈ£. ÇöÀç ÆÄÀÏÀÌ include³ª require·Î Æ÷ÇÔµÈ ÆÄÀÏÀÌ¶ó¸é, includeÇÔ

¼öµîÀ» È£ÃâÇÑ °Î¸ð ÆÄÀÏÀÌ ¾Æ´Ñ, Æ÷ÇÔµÈ ÆÄÀÏ³»¿¡¼-ÀÇ ¶óÀÎ ¹øÈ£¸¦ °¡Áø´Ù.

PHP_VERSION
ÇöÀç »ç¿ëÁßÀÎ PHP ÆÄ¼-ÀÇ ¹öÀü. (¿¹ : '3.0.8 - dev')

PHP_OS
PHP ÆÄ¼-°¡ ½ÇÇàµÇ°í ÀÖÀÀ OSÀÇ À̸§. (¿¹ : 'Linux')

TRUE
Âü(TRUE) °ª

FALSE
°ÅÁþ(FALSE) °ª

E_ERROR
¹®¹ý ¿À·ù° ¾Æ´Ñ °¹±¸° °Ò°¡´ÉÇÑ ¿¡·¯¸¦ Ç¥½ÃÇÑ´Ù.

E_WARNING
PHP°¡ ¾î¶² °è¼Ó ¼öÇà °¡´ÉÇÑ ¿À·ù¸¦ °¨Áö ÇßÀ½À» Ç¥½ÃÇÑ´Ù. ¿¹¸¦µé¾î **ereg()**¿¡ regexp°¡ ÀßºøµÈ °æ¿ì µîÀÌ´Ù.

E_PARSE
ÆÄ¼-°¡ ½ºÅ©¸³Æ® ÆÄÀÌ¿¡¼- ¹®¹ýÀûÀ¸·Î ÀßºøµÈ ¸í·ÉÀ» ¸³- °æ¿ìÀÌ´Ù. °¹±¸´Â °Ò°¡´ÉÏ´Ù.

E_NOTICE
¿¡·¯´Â ¾Æ´ÏÁö¸¸ ¾î¶² ¾Ë·ÁÁú »çÇ×ÀÌ ÀÖÀ½À» ³ªÅ¸³½´Ù. ½ÇÇàÀº °è¼ÓµÈ´Ù. ¿¹¸¦µé¾î hash ÀÎµ¦½º¿¡ µû¿ÈÇ¥°¡ ¾ø´Â ¹®ÀÚ¿-ÀÌ »ç¿ëµÇ¾ú´Ù´ØÁö, ÀÌÀü¿¡ ¼±¾ðµÇÁö ¾ÊÀº °¯¼öÀÇ °ªÀ» ÀÐ´Â´Ù´ØÁö ÇÏ´Â °æ¿ìÀÌ´Ù.

E_* »ó¼ö´Â ´ë°³ error_reporting() ÇÔ¼ö¿¡¼- reporting ·¹º§À» Á¤ÇÒ ¶§ ÈçÈ÷ »ç¿ëµÈ´Ù.

ÁÖÀÇÇÒ Á¡Àº »ó¼ö´Â C¿¡¼-ÀÇ macro °ªÀº °ÍÀÌ ¾Æ´Ï´Ù. ´ÜÁö scalar °ªÀ» °¡Áú ¼ö ÀÖÀ» »ÓÀÌ´Ù.

¿©·¯°ÐÀº **define()**ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© Ãß°¡ÀûÀÎ »ó¼ö¸¦ Á¤ÀÇÒ ¼ö ÀÖ´Ù.
Example 6 - 1. Defining Constants

```
<?php
    define("CONSTANT", "Hello world.");
    echo CONSTANT; // outputs "Hello world."
?>
```

Example 6 - 2. Using __FILE__ and __LINE__

```
<?php
    function report_error($file, $line, $message) {
        echo "An error occured in $file on line $line: $message.";
    }

    report_error(__FILE__,__LINE__, "Something went wrong!");
?>
```

Ç¥Çö½Ä (Expressions)

Ç¥Çö½ÄÀº PHP¿¡¼- ¸Å¿ì Áß¿äÇÑ ³»¿ëÀ̸ç PHP¿¡¼- ¿©·¯°ÐÀÌ ÀÛ¼ºÇÏ´Â °ÅÀÇ ¸ðµç °ÍÀº Ç¥Çö½ÄÀÌ´Ù. °£´ÜÈ÷ Ç¥Çö½ÄÀÇ Á¤ÀǸ¦ ¾ö¹ÐÇÏ°Ô ¸»ÇÑ´Ù¸é "°ªÀ» °¡Áö°í ÀÖ´Â ¸ðµç °Í"("anything that has a value")ÀÌ´Ù.

°£´ÜÇÑ ¿¹Á¦·Î "$a = 5"¿Í °°Àº °ÍÀÌ ÀÖ´Ù. ÀÌ°ÍÀº 5¶ó´Â °ªÀ» $a¶ó´Â °¯¼ö¿¡ ´ëÀÛÇÏ´Â ¹®ÀåÀÌ´Ù. ¿©±â¼- 5¶ó´Â Ç¥Çö½ÄÀº 5ÀÇ °ªÀÌ¶ó´Â ÀÇ¹Ì¸¦ µÈ´Ù. À§ÀÇ ¹®Åå ´ÙÀ½¿¡ "$b = $a"¶ó´Â ¹®ÅåÀÌ ÀÖ´Ù¸é ¿©±â¼- $a¶ó´Â Ç¥Çö½ÄÀº ±× °¯¼ö¸¦ °¡Áø 5¶ó´Â °ªÀ» Ç¥ÇöÇÑ´Ù.

Á¶±Ý °¹ÀâÇÑ °æ¿ì´Â ÇÔ¼öÀÌ´Ù. ´ÙÀ½ ÇÔ¼ö¸¦ °¸ÀÚ. :

```
function foo()
{
    return 5;
}
```

$c = 5¶ó´Â ¹®Åå¿¡¼- foo()´Â 5¸¦ µ¹·ÁÀÙ °ÍÀÌ±â ¶§¹®¿¡ 'foo()'¶ó´Â Ç¥Çö½ÄÀº 5¸¦ Ç¥ÇöÇÑ´Ù°í ¸»ÇÒ ¼ö ÀÖ´Ù.

PHP´Â Á¤¼ö°ª(integer values), ½Ç¼ö°ª(floating point values), ¹®ÀÚ¿-(string values)ÀÇ 3°¡Áö ½ºÄ®¶ó typeÀ» °¡Áö°í ÀÖ´Ù. (½ºÄ®¶ó °ªÀÌ¶õ ´õ ÀÌ»ó ÀÚÀº °ªÀ¸·Î ÂÉ°³Áú ¼ö ¾ø´Â °ªÀ» ¸»ÇÑ´Ù.) ¶ÇÇÑ PHP´Â ¹è¿-(array)°ú °´Ã¼(object)ÀÇ 2°¡Áö È¥ÇÕ(composite, non - scalar) typeÀ» Áö¿øÇÑ´Ù. ÀÌ typeµéÀº °¯¼ö¿¡ °ªÀ» ÇÒ´çÇÒ ¼ö ÀÖ°í, ÇÔ¼ö¿¡ ÀÇÇØ °ªÀ» µ¹·ÁÀÙ ¼ö ÀÖ´Ù.

¿©±â±îÁö´Â PHP/FI 2 »ç¿ëÀÚµéÀº ¾Æ¹«°Íµµ ¹Ù²îÁö ¾Ê¾Ò´Ù°í ´À³¥ °ÍÀÌ´Ù. PHP3´Â °ÅÀÇ ¸ðµç °ÍÀÌ Ç¥Çö½ÄÀ¸·Î Ç¥ÇöµÈ´Ù ´Â Á¡¿¡¼, Ç¥Çö ÁßÉ ¾ð¾î (expression-oriented language)¶ó°í ºÒ¸± ¼ö ÀÖ´Ù. Ç¥Çö ÁßÉÀÇ ÁÁÀº ¿¹´Â ´ëÀÛ¹®ÀÌ´Ù. PHP¿¡¼´Â C¿Í °°ÀÌ ´ëÂÛ¹®µµ ÇÏ³ªÀÇ Ç¥Çö½ÄÀ¸·Î °»´Ù. µû¶ó¼ '$a=5; $b=5;'´Â '$a = ($b=5);' È¤Àº '$a = $b = 5;' ·Î Ç¥Çö °¡´ÉÏ´Ù.

Ç¥Çö ÁßÉÀÇ ´Ù¸¥ ÁÁÀº ¿¹·Î´Â ++¿Í --ÀÌ´Ù. In PHP/FI 2¿¡¼´Â '$a++'¿Í °°Àº ¹®ÀåÀº ¾Æ¹« °³µµ °¡ÁöÁö ¾Ê¾Ò´Ù. Áï, Ç¥Çö½ÄÀÌ ¾Æ´Ï¾ú´Ù. µû¶ó¼ ÀÌ °³À» ´Ù¸¥ °÷¿¡ ´ëÀÛÇÏ´Â °ÍÀº °Ò¡¡´ÉÇß´Ù. ±×·¯³ª PHP 3¿¡¼´Â ÀÌ°ÍÀÌ °¡´ÉÇØÁ³´Ù. C¿¡¼¿Í °°ÀÌ ++/--¸¦ »ç¿ëÇÏ´Â ¹æ¹ý¿¡´Â ¾Õ¿¡ ºÙÀÌ´Â °Í°ú µÚ¿¡ ºÙÀÌ´Â °Í µÎ°¡Áö°¡ ÀÖ´Ù. µÑ ¸ðµÎ °¯¼öÀÇ °³ÀÌ Áõ°¡µÈ´Ù ´Â Á¡Àº °°À¸³ª, '++$variable'À° ±× °¯¼öÀÇ Áõ°¡µÈ °³ÀÌ Ç¥Çö½ÄÀÇ °³ÀÌ µÇ°í, '$variable++'´Â °¯¼öÀÇ Áõ°¡µÇ±â ÀüÀÇ °³ÀÌ Ç¥Çö½ÄÀÇ °³ÀÌ µÈ´Ù.

°ñ±³ Ç¥Çö½ÄÀº 0ÀÌ³ª 1ÀÇ °³À» °¡Áø´Ù. 0Àº °ÎÁ¤(FALSE)À» 1Àº ±àÁ¤(TRUE)¸¦ ³ªÅ¸³½´Ù. PHP´Â > (bigger than), >= (bigger than or equal to), == (equal), < (smaller than), <= (smaller than or equal to)¸¦ Áö¿øÇÑ´Ù.

¸¶Áö¸· ¿¹´Â °áÇÕµÈ ¿¬»ê-´ëÂô Ç¥Çö½Ä(combined operator-assignment expression)ÀÌ´Ù. ¹¹ °¹ÀâÇÑ ¾ê±â´Â ¾Æ´Ï°í '$a = $a + 3'À° Ç¥Çö½ÄÀ» '$a += 3'À¸·Î Ç¥ÇöÇÒ ¼ö ÀÖ´Â °ÍÀÌ´Ù. -=, *=, /=, .= µîµµ »ç¿ëÇÒ ¼ö ÀÖ´Ù.

´ÙÀ½¿¹¸¦ °¸¸é µµ¿òÀÌ µÉ °ÍÀÌ´Ù. :

```
function double($i)
{
        return $i*2;
}
$b = $a = 5;          /* $a¿Í $b ¿¡ 5 ´ëÀÔ */
$c = $a++;            /* post-increment, $c ¿¡´Â $aÀÇ ¿ø·¡ °ªÀÎ 5°¡ ´ëÀÔµÇ°í $a´Â 6ÀÌ µÈ´Ù */
$e = $d = ++$b;       /* pre-increment, $d¿Í $e´Â Áõ°¡µÈ $bÀÇ °ª 6ÀÌ ´ëÀÔµÈ´Ù. */

/* ¿©±âº´ÎÁ $d¿Í $eÀÇ °ªÀº 6ÀÌ´Ù. */
$f = double($d++);    /* $f´Â $d°¡ Áõ°¡µÇ±â ÀüÀÇ °ªÀÎ 6À» µÎ ¹èÇÑ °ªÀ» °¡Áø´Ù. 2*6 = 12 */
$g = double($++e);    /* $g´Â $e°¡ Áõ°¡µÈ ÈÄÀÇ °ªÀÎ 7À» µÎ¹èÇÑ°ªÀ» °¡Áø´Ù. 2*7 = 14 */
$h = $g += 10;        /* $g´Â 14¿¡ 10À» ´õÇØ 24ÀÇ °ªÀ» °¡Áø ÈÄ ±× °ªÀ» $h¿¡ ´ëÀÔÇÑ´Ù. µÑ´Ù 24 */
```

±×·¯³ª, ÇÏ³ªÀÇ Ç¥Çö½ÄÀÌ ÇÏ³ªÀÇ ¹®ÀåÀº ¼Æ´Ï´Ù. Ç¥Çö½ÄÀÌ ¼¼¹ÌÄÝ·Ð ';'À¸·Î ³¡³ª¾ß ÇÏ³ªÀÇ ¹®ÀåÀÌ µÇ´Â °ÍÀÌ´Ù. Áï '$b=$a=5;'¿¡¼ $a=5´Â Ç¥Çö½ÄÀÌÁö¸¸ ¹®ÀåÀº ¾Æ´Ï´Ù.

PHP¿¡¼ Âü/°ÅÁþÀÇ ÆÇ´ÜÀº Perl°ú °ñ½ÁÇÏ´Ù. 0ÀÌ ¾Æ´Ñ ¼ýÀÚ´Â ¸ðµÎ TRUEÀÌ°í, 0Àº FALSEÀÌ´Ù. °ó ¹®ÀÚ¿ÀÌ³ª ¹®ÀÚ¿ "0"Àº FALSEÀÌ´Ù. ´Ù¸¥ ¸ðµç ¹®ÀÚ¿Àº TRUEÀÌ´Ù. ¹è¿ÀÌ³ª °´Ã¼´Â ¿ø¼Ò¡ ÇÏ³ªµµ ¾ø´Ù¸é FALSE¸¦ ÀÇ¹ÌÇÏ°í, ³ª¸ÓÁöÀÇ °æ¿ì´Â TRUEÀÌ´Ù.

¼ÕÀ¸·Î ÀÌ ¸Å´º¾ó¿¡¼ 'expr'ÀÌ¶ó°í ¾²¿©Áø °ÎºÐÀº PHP3ÀÇ ¿Ã¹Ù¸¥ Ç¥Çö½ÄÀÌ¶ó´Â ÀÇ¹ÌÀÌ´Ù.

IF

PHPÀÇ IF¹®Àº C¿Í °ñ½ÁÇÏ´Ù.

```
if (expr)
  statement
```

exprÀ» Æò°¡Çß¿© TRUEÀÌ¸é statement¸¦ ½ÇÇàÇÑ´Ù. FALSEÀÌ¸é ¹«½ÃÇÑ´Ù.

´ÙÀ½ ¿¹Á¦´Â $a°¡ $b°¸´Ù Å©¸é "a is bigger than b"¸¦ Ãâ·ÂÇÑ´Ù.

```
if ($a > $b)
  print "a is bigger than b";
```

¿©·¯ ¹®ÀåÀ» ½ÇÇà½ÃÅ°·Á¸é ÁßºýÈ£({ })¸¦ »ç¿ëÇÏ¿© °¹¹®À¸·Î ¸¸µé¸éÈ´Ù. :

```
if ($a>$b) {
  print "a is bigger than b";
  $b = $a;
}
```

ELSE

ELSE´Â IF¹®ÀÇ Æò°¡½ÄÀÌ FALSEÀÏ °æ¿ì ½ÇÇàÇÏ´Â ¹®ÀåÀ» ÁöÁ¤ÇÑ´Ù. :

```
if ($a>$b) {
    print "a is bigger than b";
} else {
    print "a is NOT bigger than b";
}
```

ELSEIF

ELSEIF´Â ELSE¿Í IF¸¦ ÇÕÃÄ³õÀº °Í°ú °°´Ù. :

```
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a is equal to b";
} else {
    print "a is smaller than b";
}
```

ÇÑ IF¹®¿¡´Â ¿©·¯°³ÀÇ ELSEIF¹®ÀÌ ÀÖÀ» ¼ö ÀÖ´Ù. ELSEIF¹®Àº ¼ø¼-´ë·Î Æò°¡ÇÏ¿© TRUEÀÎ °÷À» ½ÇÇàÇÑ´Ù. Áï, ¾î¶² ELSEIF¹®ÀÌ ½ÇÇà µÇ·Á¸é IF¹®ÀÇ Æò°¡½Ä°ú ±× ¾ÕÀÇ ¸ðµç ELSEIF¹®ÀÇ Æò°¡½ÄÀÌ FALSEÀÌ¾î¾ß ÇÑ´Ù. ÀÌ°ÍÀº 'else if'¶ó°í µÎ °³ÀÇ ¹®ÀåÀ¸·Î ½áµµ µÈ´Ù(¹®¹ýÀûÀÎ ÀÇ¹Ì´Â ¾à°£ ´Ù¸£Áö¸¸).

IFÀÇ ´Ù¸¥ Ç¥Çö (Alternative syntax for IF statements): IF(): ... ENDIF;

PHP 3´Â Áß°ýÈ£({ })¸¦ ¾²´Â ´ë½Å IF(expr)µÚ¿¡ ÄÝ·Ð(:)À» Âï°í, ÇÏ³ª ÀÌ»óÀÇ ¹®ÀåÀ» ³ª¿-ÇÑ ÈÄ¿¡ ENDIF;·Î ³¡³»´Â ¹æ¹ýÀ» Á¦°øÇÑ´Ù. ÀÌ ¹æ¹ýÀº Æ¯È÷ IF¹® ¾È¿¡ HTML °í·°À» »ðÀÔÇÏ´Âµ¥ À¯¿ëÇÏ°Ô »ç¿ëµÉ ¼ö ÀÖ´Ù. ´ÙÀ½ ¿¹¸¦ º¸ÀÚ. :

```
<?php if ($a==5): ?>
A = 5
<?php endif; ?>
```

À§ÀÇ ¿¹¿¡¼- "A = 5"¶ó´Â HTML °í·ÏÀÌ IF¹® ¾È¿¡ »ç¿ëµÇ°í ÀÖ´Ù. À§ÀÇ HTML °í·ÏÀº $a°¡ 5ÀÏ °æ¿ì¿¡¸¸ Ç¥½ÃµÈ´Ù.

´ÙÀ½°ú °°ÀÌ ELSE¿Í ELSEIF (expr)µµ »ç¿ëÇÒ ¼ö ÀÖ´Ù. :

```
if ($a==5):
    print "a equals to 5";
    print "...";
elseif ($a==6):
    print "a equals to 6";
    print "!!!";
else
    print "a is not 5 nor 6";
endif;
```

WHILE

WHILE ·çÇÁ´Â PHP 3ÀÇ °¡Àå °£´ÜÇÑ ÇüÅÂÀÌ´Ù. ÀÌ°ÍÀº C¿Í µ¿ÀÏÇÏ°Ô ÀÛµ¿ÇÑ´Ù. ±â» ÇüÅÂ´Â ´ÙÀ½°ú °°´Ù. :

```
    WHILE(expr) statement
```

IF¹®°ú ¸¶Âù°¡Áö·Î ´ÙÀ½°ú °°Àº ¹®Àå ÇüÅÂ·Î Áß°ýÈ£({ })¸¦ »ç¿ëÇÏÁö ¾Ê´Â ¹æ¹ýµµ ÀÖ´Ù. :.

```
    WHILE(expr): statement ... ENDWHILE;
```

´ÙÀ½µÎ°³ÀÇ ¿¹Á¦´Â µ¿ÀÏÇÑ ¿¹Á¦ÀÌ´Ù. 1°ÎÅÍ 10±îÁö Ãâ·ÂÇÑ´Ù.:

```
/* example 1 */
$i=1;
while ($i<=10) {
    print $i++;  /* the printed value would be $i before the increment (post-increment) */
}
/* example 2 */
$i=1;
while ($i<=10):
    print $i;
    $i++;
endwhile;
```

DO..WHILE

DO..WHILE ·çÇÁ´Â °ñ±³½ÃÀÌ ¾ÕÀÌ ¾Æ´Ñ ¸ÚÇ µÚ¿¡ ÀÖ´Ù´Â Á¡À» Á¦¿ÜÇÏ¸é WHILE ·çÇÁ¿Í °ñ½ÁÇÏ´Ù. µû¶ó¼- WHILE Á¶°Ç ÀýÀº DO ¹®ÅåÀÌ ½ÇÇàµÈ ÈÄ¿¡ Æò°¡µÇ¹Ç·Î, DO ÀÌÈÄÀÇ ¹®Åå°º ¹«Á¶°Ç ÇÑ¹øÀº ½ÇÇàµÈ´Ù.

´ÙÀ½°ú °°Àº DO..WHILE ·çÇÁ¿¡¼-µµ ÇÑ¹øÀº Ãâ·ÂÀÌ µÈ´Ù. :

```
$i = 0;
do {
```

```
    print $i;
} while ($i>0);
```

FOR

FOR´Â PHPÀÇ °¡Àå ºÎÀâÇÑ ·çÇÁÀÌ´Ù. ÀÌ°ÍÀº C¿Í À¯»çÇÏ´Ù. FOR ·çÇÁÀÇ ÇüÅÂ´Â ´ÙÀ½°ú °°´Ù. :

```
    FOR (expr1; expr2; expr3) statement
```

Ã¹ ¹øÂ° Ç¥Çö½Ä(expr1)À° ·çÇÁ¸¦ ½ÃÀÛÇÒ ¶§ ¹«Á¶°Ç ÇÑ¹ø Æò°¡(½ÇÇà)µÈ´Ù.

¸Å ¹Ý°¹ÀÇ ½ÃÀÛ ¶§¸¶´Ù expr2°¡ Æò°¡µÈ´Ù. ¸¸¾à ÀÌ°ÍÀÌ TRUE¸é ·çÇÁ´Â °è¼ÓµÇ°í statement°¡ ½ÇÇàµÈ´Ù. expr2°¡ FALSEÀÌ¸é ·çÇÁ´Â Á¾·áµÈ´Ù.

¸Å ¹Ý°¹ÀÌ ³¡³¯ ¶§ expr3°¡ Æò°¡(½ÇÇà)µÈ´Ù.

°¢ Æò°¡½ÄÀº °ñ¿öµÑ ¼ö ÀÖ´Ù.expr2°¡ °ñ¾îÀÖÀ¸¸é ¹«ÇÑ ·çÇÁ¸¦ ¶æÇÑ´Ù. (PHP´Â C¿Í °°ÀÌ °ñ¾îÀÖÀ¸¸é TRUE·Î ÀÎ½ÄÇÑ´Ù.) ÀÌ°Ç °°·Î ÁÁÀº ¹æ¹ýÀÌ ¾Æ´ÏÁö¸¸, Á¾Á¾ BREAK¸¦ »ç¿ëÇÏ¿© Á¾·áÇÏ´Â ¹æ¹ýµµ ÀÖ´Ù.

´ÙÀ½ ¿¹´Â 1¿¡¼ 10±îÁö Ãâ·ÂÇÏ´Â ¿¹ÀÌ´Ù. :

```
/* example 1 */
for ($i=1; $i<=10; $i++) {
    print $i;
}
/* example 2 */
for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}
/* example 3 */
$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}
/* example 4 */
for ($i=1; $i<=10; print $i, $i++) ;
```

¹°·Đ Ã³À½°ÍÀÌ °¡Àå ÁÁ¾Æ¸¸ÀÎ´Ù. ±×·¯³ª ³ª¸ÓÁö¸µ °¡´ÉÇÏ´Ù´Â °ÍÀ» ¾Ë¾Æ¾ß ÇÑ´Ù.

PHP´Â FOR ·çÇÁ¿¡ ´ëÇØ¼µµ ´ÙÀ½°ú °°Àº "colon syntax"¸¦ Áö¿øÇÑ´Ù.

```
    FOR (expr1; expr2; expr3): statement; ...; endfor;
```

ÀÏ°Î ´Ù¸¥ ¾ð¾îµéÀº ¹è¿À» Å½»öÇÏ±â À§ÇÏ¿© foreach ¹®ÀåÀ» Á¦°øÇÏ±âµµ ÇÑ´Ù. ±×·¯³ª PHP´Â ÀÌ¸¦ À§ÇØ **list()¿Í each()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© while ·çÇÁ·Î ÇØ°áÇÑ´Ù. ÀÌ ÇÔ¼öµé¿¡ ´ëÇØ¼´Â ¿¹Á¦¸¦ ÂüÁ¶ÇÏÀÚ.

BREAK

BREAK´Â ÇöÀç ·çÇÁ¿¡¼ °üÁ® ³ª°¡´Â ¸í·ÉÀÌ´Ù.

```
$i = 0;
while ($i < 10) {
    if ($arr[$i] == "stop") {
        break;
    }
    $i++;
}
```

CONTINUE

CONTINUE ´Â ÇöÀç ·çÇÁÀÇ Ã³À½¸·Î °¡µµ·Ï ÇÏ´Â ¸í·ÉÀÌ´Ù.

```
while (list($key,$value) = each($arr)) {
    if ($key % 2) { // skip even members
        continue;
    }
```

```
    do_something_odd($value);
}
```

SWITCH

SWITCH¹®Àº µ¿ÀÏÇÑ Ç¥Çö½ÄÀÇ IF¹®ÀÇ ³ª¿-°ú °ñ½ÁÇÏ´Ù. ´ÙÀ½ ¿¹´Â µ¿ÀÏÇÑ °ÍÀº IF¹®°ú SWITCH¹®À¸·Î Ç¥ÇöÇÑ °ÍÀÌ´Ù. :

```
/* example 1 */
if ($i == 0) {
    print "i equals 0";
}
if ($i == 1) {
    print "i equals 1";
}
if ($i == 2) {
    print "i equals 2";
}

/* example 2 */
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
```

SWITCH¹®Àº ¹®Àå ´ÜÀ§·Î ½ÇÇàµÈ´Ù. SWITCH¿¡ ÀÖ´Â Æò°¡½Ä°ú ÀÏÄ¡ÇÏ´Â CASE¹®Àº Ã£¾Æ ±× ÀÌÈÄ SWITCHº¸í·°ÀÌ ³¡³¯ ¶§ ±îÁö ½ÇÇàÇÑ´Ù. µû¶ó¼- ¿øÇÏ´Â °æ¿ì BREAK·Î ½ÇÇàÀ» ÁßÁö½ÃÅ³ ÇÊ¿ä° ¡ ÀÖ´Ù. ´ÙÀ½ ¿¹¸¦ º¸ÀÚ. :

```
/* example 3 */
switch ($i) {
  case 0:
    print "i equals 0";
  case 1:
    print "i equals 1";
  case 2:
    print "i equals 2";
}
```

¿©±â¼- $i°¡ 0ÀÌ¸é ¸ðµç print¹®Àº ½ÇÇàÇÒ °ÍÀÌ´Ù. ¸¸¾à $i°¡ 1ÀÌ¸é ¶¼ö· µÎ°³ÀÇ print¹®Àº ½ÇÇàÇÑ´Ù. µû¶ó¼- ¿©·¯°ÐÀÌ ¿ø ÇÏ´Â °á°ú¸¦ ¾òÀ¸·Á¸é BREAK¹®Àº ÀØÁö ¾Ê´Â °ÍÀÌ ÁßÇäÇÑ´Ù.

Æ¯°°ÇÑ case·Î default case°¡ ÀÖ´Ù. ÀÌ °ÍÀº ´Ù¥ ¾î¶² case¿¡µµ ¸ÂÁö ¾Ê´Â °æ¿ì¸¦ ÀÇ¹ÌÇÑ´Ù. ¿¹¸¦ µé¾î :

```
/* example 4 */
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}
```

´Ù¥ ÁßÇäÇÑ Á¡Àº CASE Ç¥Çö½Ä¿¡ ´Â Á¤ö, ½Ç¼ö, ¹®ÀÚ¿-ÀÇ ½°Ä®¸® Å¸ÀÔ¸·Î Æò°¡µÇ´Â ¾î¶² Ç¥Çö½ÄÀÌ¿Íµµ µÈ´Ù´Â °ÍÀÌ´Ù. ¹è¿-ÀÌ³ª °´Ã¼µµ ¹®Á¦¸¸ ÀÏÄ¸Å°Áö´Â ¾Ê´Â´Ù. ±×·¯³ª ±×°ÍÀº ¹®Àå¿¡¼- ÀÇ¹Ì°¡ ¾ø´Ù.

REQUIRE

REQUIRE¹®Àº C preprocessorÀÇ #include¿Í °ñ½ÁÇÏ°Ô, ÀÚ½ÅÀ» ÁöÁ¤µÈ ÆÄÀÏ·Î ´ëÃ¼ÇÑ´Ù.

ÀÌ°ÍÀº require()¹®À» ·çÇÁ ±¸Á¶ ¾È¿¡ µÎ¾î, ¸Å¹ø ´Ù¸¥ ÆÄÀÏÀ» ÀÐ¾îµéÀÏ ¼ö ¾ø´Ù´Â °ÍÀ» ÀÇ¹ÌÇÑ´Ù. ÀÌ·± µ¿ÀÛÀ» À§ÇØ¼- ´Â INCLUDE ¹®À» »ç¿ëÇÏ¶ó.

```
require('header.inc');
```

INCLUDE

INCLUDE¹®À° ÁöÁ¤ÇÑ ÆÄÀÏÀ» ÀÐ°í ½ÇÇàÇÑ´Ù.

ÀÌ µ¿ÀÛÀº ½ÇÇàÁß INCLUDE ¹®À» ¸¸³¯ ¶§ ¸¸´Ù ÀÏ¾î³-´Ù. µû¶ó¼- ¿©·¯ºÐÀº INCLUDE¹®À» ·çÇÁ ±¸Á¶ ¾È¿¡ µÎ¾î ¸Å¹ø ´Ù¸¥ ÆÄÀÏÀ» ÀÐ¾î µéÀÌµµ·Ï ÇÒ ¼ö ÀÖ´Ù.

```
$files = array('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
  include($files[$i]);
}
```

include()´Â ÀÌ ¹®ÀåÀ» ¸¸³¯ ¶§ ¸¶´Ù ¸Å¹ø ÀçÆò°¡µÇ¾î Àç½ÇÇàµÈ´Ù´Â Á¡¿¡¼- **require()**¿Í ´Ù¸£´Ù. ¹Ý¸é¿¡ **require()**¹®À° ÁöÁ¤µÈ ÆÄÀÏÀÇ ³»¿ëÀÌ ½ÇÇàµÇ´Â°¡¿¡ °ü°è¾øÀÌ(¿¹¸¦µé¾î if ¹® ¾È¿¡ µé¾îÀÖ°í »óÅÂ°¡ °ÅÁþÀÎ °æ¿ì¿¡µµ), ÀÌ ¹®ÀåÀ» Ã³À½ ¸¸³¯Àº ¶§ ÁöÁ¤µÈ ÆÄÀÏ·Î ´ëÄ¡µÈ´Ù.

include()´Â Æ¯°ÇÑ ±¸Á¶ÀÌ¹Ç·Î, ¸¸¾à ÀÌ°ÍÀÌ Á¶ÀüÀý ¾È¿¡ ³õ¿©ÀÖ´Ù¸é ¹Ýµå½Â {}(statement block)À¸·Î µÑ·¯½Î¾ß ÇÑ´Ù.

```
/* This is WRONG and will not work as desired. */
if ($condition)
    include($file);
else
    include($other);
/* This is CORRECT. */
if ($condition) {
    include($file);
} else {
    include($other);
}
```

ÁöÁ¤µÈ ÆÄÀÏÀÌ Æò°¡µÉ ¶§, ÆÄ¼-´Â "HTML - mode"¿¡¼- ½ÃÀÛÇÑ´Ù. µû¶ó¼- PHP ¹®ÀåÀº PHP ½ÃÀÛ ÅÅ(<?)À» ÁÖ°í ½ÃÀÛÇÏ¿©¾ß ÇÑ´Ù.

See also **readfile()**, **require()**, **virtual()**.

FUNCTION

ÇÔ¼ö´Â ´ÙÀ½°ú °°ÀÌ Á¤ÀÇÇÑ´Ù. :

```
function foo( $arg_1, $arg_2, ..., $arg_n ) {
   echo "Example function.\n";
   return $retval;
}
```

ÇÔ¼ö ¾È¿¡´Â ´Ù¸¥ ÇÔ¼ö³ª classÀÇ ¼±¾ð µîÀ» Æ÷ÇÔÑ ¸ðµç °¡´ÉÑ PHP3 ÄÚµå°¡ »ç¿ëµÉ ¼ö ÀÖ´Ù.

Returning values

ÇÔ¼ö´Â return ¹®À» ÅëÇØ ÇÔ¼ö°ªÀ» µ¹·ÁÙ ¼ö ÀÖ´Ù. list¿Í object¸¦ Æ÷ÇÔÑ ¾î¶² typeµµ µ¹·ÁÚ ¼ö ÀÖ´Ù.

```
function my_sqrt( $num ) {
   return $num * $num;
}
echo my_sqrt( 4 );   // outputs '16'.
```

¿©·¯°ªÀ» µ¹·ÁÖ´Â ÀÏÀº ÇÒ ¼ö ¾ø´Ù. ±×·¯³ª list¸¦ µ¹·ÁÙ¸·Î½á ºñ½ÁÑ ÀÏÀ» ÇÒ ¼ö ÀÖ´Ù. :

```
function foo() {
   return array( 0, 1, 2 );
}
list( $zero, $one, $two ) = foo();
```

Arguments

argument list¸¦ ÅëÇØ ÇÔ¼ö¿¡ ¾î¶² Á¤º¸¸¦ ³Ñ°ÜÁÙ ¼ö ÀÖ´Ù. ÀÌ argument list´Â ½°Ç¥(,)·Î ³ªÀ§¾îÁø º¯¼ö³ª »ó¼öÀÇ listÀÌ´Ù.

PHP3´Â passing by value(±â°»ÀûÀ¸·Î ÀÌ°ÍÀ» »ç¿ë)¿Í passing by reference, default argument valuesÀÇ 3°¡Áö ¹æ¹ýÀ» Á¦°øÇÑ´Ù. °¡º¯ ±æÀÌ(Variable - length) argument list´Â Á¦°µÇÁö ¾Ê´Â´Ù. ±×·¯³ª ¹è¿-À» ÅëÇØ Àü´ÞÇÑ´Ù¸é °ñ½ÁÇÑ È¿°ú¸¦ ³¾ ¼ö ÀÖ´Ù.

Passing by reference

±â°»ÀûÀ¸·Î ÇÔ¼öÀÇ ÀÎ¼ö(argument)µéÀº °ªÀ¸·Î Àü´ÞµÈ´Ù(passed by value). ÇÔ¼ö³»¿¡¼ °¡Èˉ½ÃÅ² °ªÀ» ±×´ë·Î À¯ÁöÇÏ·Á ¸é pass by reference·Î ÀÎ¼ö¸¦ ³Ñ°Ü¾ß ÇÑ´Ù.

¿©·¯°ÐÀÌ ¾î¶² ÇÔ¼öÀÇ ÀÎ¼ö¸¦ Ç×»ó pass by reference·Î ³Ñ±â·Á ÇÑ´Ù¸é, ¿©·¯°ÐÀº ÇÔ¼ö¸¦ ¼±¾ðÇÒ ¶§ ampersand(&)¸¦ ÀÎ¼öÀÇ ¾Õ¿¡ °Ù¿©ÁÖ¸é µÈ´Ù. :

```
function foo( &$bar ) {
   $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo2( $str );
echo $str;    // outputs 'This is a string, and something extra.'
```

¸¸¾à ±â°»À° by value ·Î ÇÏÁö¸¸ ÇÊ¿ä¿¡ µû¶ó by reference·Î È£ÃâÇÏ°í ½Í´Ù¸é ÇÔ¼ö È£Ãâ ½Ã¿¡ ÀÎ¼öÀÇ ¾Õ¿¡ &¸¦ °ÙÀÌ¸é µÈ´Ù. :

```
function foo( $bar ) {
   $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo2( $str );
echo $str;    // outputs 'This is a string, '
foo2( &$str );
echo $str;    // outputs 'This is a string, and something extra.'
```

Default values

½ºÄ®¶ó ÀÎ¼ö´Â ´ÙÀ½°ú °°ÀÌ C++ °ú ºñ½ÁÇÏ°Ô ±â°»°ªÀ» Á¤ÇØÁÙ ¼ö ÀÖ´Ù. :

```
function makecoffee( $type = "cappucino" ) {
   echo "Making a cup of $type.\n";
}
echo makecoffee();
echo makecoffee( "espresso" );
```

À§ÀÇ ´ÜÆíÀÇ ½ÇÇà °á°ú´Â ´ÙÀ½°ú °°´Ù :

```
Making a cup of cappucino.
Making a cup of espresso.
```

default argument¸¦ »ç¿ëÇÒ ¶§, default°¡ µÇ´Â ÀÎ¼öµéÀº non-defaultÀÎ ÀÎ¼öµéº¸´Ù ¿À¸¥ÂÊ¿¡ À§Ä¡ÇØ¾ß ÇÑ´Ù. ±×·¸Áö ¾ÊÀ¸¸é ¿øÇÏ´Â °á°ú°¡ ³ª¿ÀÁö ¾Ê´Â´Ù. ´ÙÀ½À» º¸ÀÚ. :

```
function makeyogurt( $type = "acidophilus", $flavour ) {
   return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt( "raspberry" );   // won't work as expected
```

À§ ÄÚµåÀÇ ½ÇÇà °á°ú´Â ´ÙÀ½°ú °°´Ù :

```
Warning: Missing argument 2 in call to makeyogurt() in /usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

±×·¯¸é ÀÌÁ¦ À§ÀÇ °Í°ú ¾Æ·¡°ÍÀ» º¯±³ÇØ º¸ÀÚ. :

```
function makeyogurt( $flavour, $type = "acidophilus" ) {
   return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt( "raspberry" );   // works as expected
```

ÀÌ ¿¹Á¦ÀÇ ½ÇÇà °á°ú´Â ´ÙÀ½°ú °°´Ù. :

```
Making a bowl of acidophilus raspberry.
```

OLD_FUNCTION

OLD_FUNCTION ¹®ÀåÀº PHP/FI2¿¡¼ µ¿ÀÏÇÑ ÇÔ¼ö »ç¿ë¹ýÀ» Á¦°øÇÑ´Ù. (function´ë½Å old_functionÀ» »ç¿ëÇÑ´Ù´Â Á¡ÀÌ Á¦¿ÜÇÏ°í)

ÀÌ°ÍÀ» »ç¿ëÇÏ´Â °ÍÀº ÁÁÁö ¾ÊÀº ¹æ¹ýÀÌ´Ù. ÀÌ°ÍÀÌ »ç¿ëµÉ ¶§´Â PHP/FI2 -> PHP3 °¯È¯±â¿¡¼ »ÓÀÌ´Ù.

OLD_FUNCTIONÀ¸·Î Á¤ÀÇµÈ ÇÔ¼öµéÀº PHPÀÇ ³»°Î ÄÚµå¿¡¼- È£ÃâµÉ ¼ö ¾ø´Ù. ÀÌ ¸»À° usort()³ª array_walk(), register_shutdown_function()°°À° ÇÔ¼ö¿¡ »ç¿ëÇÒ ¼ö ¾ø´Ù´Â ÀÇ¹ÌÀÌ´Ù. ÀÌ¸¦ ÇØ°áÇÏ±â À§ÇØ¼-´Â ÀÌ OLD_FUNCTIONÀ¸·Î ¼±¾ðµÈ ÇÔ¼ö¸¦ È£ÃâÇÏ´Â PHP3 ÇüÅÂÀÇ ÇÔ¼ö¸¦ ¸¸µé¾î »ç¿ëÇÏ´Â °ÍÀÌ´Ù.

CLASS

(¿ªÀÚÁÖ : Class ¹®ÀåÀº JavaÀÇ subsetÀÌ¶ó ÇÒ¸¸Å- µ¿ÀÏÇÏ´Ù.)

Å¬·¡½º´Â ÀÏ·ÃÀÇ º¯¼ö¿Í ÀÌ º¯¼öµéÀ» »ç¿ëÇÏ´Â ÇÔ¼öµéÀÇ ¸ðÀ½ÀÌ´Ù. Å¬·¡½º´Â ´ÙÀ½°ú °°Àº ÇüÅÂ·Î ¼±¾ðµÈ´Ù.

```
<?php
class Cart {
  var $items;  // Items in our shopping cart

  // Add $num articles of $artnr to the cart
  function add_item($artnr, $num) {
    $this->items[$artnr] += $num;
  }

  // Take $num articles of $artnr out of the cart
  function remove_item($artnr, $num) {
    if ($this->items[$artnr] > $num) {
      $this->items[$artnr] -= $num;
      return true;
    } else {
      return false;
    }
  }
}
?>
```

ÀÌ ¼±¾ðÀº Cart¶ó´Â ÀÌ¸§ÀÇ Å¬·¡½º¸¦, Ä«Æ®¿¡ µé¾îÀÖ´Â ¹°Ç°À» À§ÇÑ ÇÑ °³ÀÇ ¹è¿- º¯¼ö¿Í cart¿¡ ¹°°ÇÀ» ³Ö°Å³ª »©´Â µÎ °³ÀÇ ÇÔ¼ö·Î ±¸¼º µÇ¾î ÀÖ´Ù.

Classe´Â TypeÀ¸·Î, ½ÇÁ¦ º¯¼öµéÀÇ Ã»»çÁøÀÌ¶ó ÇÒ ¼ö ÀÖ´Ù. ¿©·¯°ÐÀº new ¿¬»êÀú¸¦ »ç¿ëÇÏ¿© ¿øÇÏ´Â typeÀÇ º¯¼ö¸¦ »ý¼ºÇÏ¿©¾ß ÇÑ´Ù.

```
$cart = new Cart;
$cart->add_item("10", 1);
```

À§¿¡ ÀÖ´Â ¿¹´Â Cart Å¬·¡½ºÀÇ $cart¶ó´Â object¸¦ ¸¸å´Â °ÍÀÌ´Ù. ÀÌ objectÀÇ add_item() ÇÔ¼ö¸¦ È£ÃâÇÏ¿© ¹°Ç° ¹øÈ£ "10"¹øÀÇ ¹°Ç° 1°³¸¦ Ä«Æ®¿¡ ³Ö´Â´Ù.

Å¬·¡½º´Â ´Ù¸¥ Å¬·¡½º È®Àå µÉ ¼ö ÀÖ´Ù. È®Àå È¤Àº ÆÄ»ýµÈ(extended or derived) Å¬·¡½º´Â base °¡ µÇ´Â Å¬·¡½ºÀÇ ¸ðµç º¯¼öµé°ú ÇÔ¼öµéÀ» ±×´ë·Î °¡Á¦Ô µÇ°í, ¿©·¯°ÐÀº ¿©±â¿¡ Ãß°¡·Î È®ÀåµÈ ¼±¾ðÀ» ÇÒ ¼ö ÀÖ´Ù. ÀÌ¸¦ À§ÇØ "extends"¶ó´Â Å°¿öµå°¡ »ç¿ëµÈ´Ù.

```
class Named_Cart extends Cart {
  var $owner;

  function set_owner($name) {
    $this->owner = $name;
  }
}
```

À§ÀÇ ¿¹´Â Cart Å¬·¡½ºÀÇ º¯¼ö¿Í ÇÔ¼ö¿¡ $owner º¯¼ö¿Í set_owner() ÇÔ¼ö¸¦ Ãß°¡ÇÑ Named_Cart¶ó´Â Å¬·¡½ºÀÇ ¼±¾ðÀ¸·Î ¿©·¯°ÐÀº ÀÌ¸§ºÙÀº Ä«Æ®(named cart)¸¦ »ç¿ëÇÏ¿© Ä«Æ®ÀÇ ÁÖÀÎÀ» ¾³Á¤ÇÏ°í Å£¾Æ¾¼ ¼ö ÀÖ´Ù. ¶ÇÇÑ ±âÁ¸ÀÇ ÀÏ¹Ý Ä«Æ®(normal cart)¿¡ ÀÖ´ø ÇÔ¼öµµ »ç¿ëÇÒ ¼ö ÀÖ´Ù.

```
$ncart = new Named_Cart;   // Create a named cart
$ncart->set_owner("kris"); // Name that cart
print $ncart->owner;       // print the cart owners name
$ncart->add_item("10", 1); // (inherited functionality from cart)
```

Å¬·¡½º »ý¿ÎÀ¿¡ ÀÖ´Â ÇÔ¼ö¿¡¼- $this ¶ó´Â º¯¼ö´Â ÀÚ±â ÀÚ½Å object¸¦ ÀÇ¹ÌÇÑ´Ù. ¿©·¯°ÐÀº $this->something ÀÇ ÇüÅÂ·Î Çö Àç objectÀÇ º¯¼ö³ª ÇÔ¼ö¸¦ »ç¿ëÏ¿©¾ß ÇÑ´Ù.

»ý¼ºÀÚ(Constructor)´Â ÇØ´ç Å¬·¡½ºÀÇ »õ ÀÎ½ºÅÏ½º(»õ·Î ¸¸µç º¯¼ö¶ó°í »ý°¢ÇØ µÎÀÚ)¸¦ ¸¸µé¶§ ÀÚµ¿ÀûÀ¸·Î ½ÇÇàµÇ´Â ÇÔ¼ö¸¦ ÀÇ¹ÌÇÑ´Ù. Å¬·¡½ºÀÇ ÀÌ¸§°ú °°Àº ÀÌ¸§ÀÇ ÇÔ¼ö¸¦ »ý¼ºÀÚ¸¦ µÈ´Ù.

```
class Auto_Cart extends Cart {
    function Auto_Cart() {
        $this->add_item("10", 1);
    }
}
```

À§ÀÇ ¿¹´Â Cart Å¬·¡½º¿¡ ¹°Ç°¹øÈ£ 10¹øÀÇ ¹°Ç°À» ÇÑ °³ ÀÚµ¿À¸·Î Ãß°¡ÇÏ´Â »ý¼ºÀÚ¸¦ Ãß°¡ÇÑ Auto_Cart¶ó´Â Å¬·¡½º¸¦ ¼±¾ðÀÌ´Ù. Auto_Cart´Â "new"·Î »ý¼º¸È´Ù. »ý¼ºÀÚ´Â ¶ÇÇÑ ÀÎ¼ö(argument)¸¦ °¡Áú ¼ö ÀÖ°í, ÀÌ ÀÎ¼öµéÀº default°ªÀ» °¡Áø ¿É ¼ÇÀ¸·Î ¼±¾ðÇÒ ¼öµµ ÀÖ´Ù. ÀÌ ±â´ÉÀº ¸Å¿ì À¯¿ëÇÏ°Ô »ç¿ëµÈ´Ù.

```
class Constructor_Cart {
    function Constructor_Cart($item = "10", $num = 1) {
        $this->add_item($item, $num);
    }
}

// Shop the same old boring stuff.

$default_cart   = new Constructor_Cart;

// Shop for real...

$different_cart = new Constructor_Cart("20", 17);
```

Chapter 7. Ç¥Çö½Ä (Expressions)

¿¬»êÀÚ (Operators)

»ê¼ú ¿¬»êÀÚ (Arithmetic Operators)

ÇÐ±³¿¡¼ ¹è¿î »ê¼ú ¿¬»ê(°¡°¨½ÂÁ¦)À» ±â¾ïÇÏ´Â°¡? ÀÌ »ê¼ú ¿¬»êÀÚ´Â ±×°Íµé°Ã³·³ µ¿ÀÛÇÑ´Ù.

Table 7 - 1. Arithmetic Operators

¿¹	ÀÌ¸§	°á°ú
$a + $b	µ¡¼ÀÀ(Addition)	$a¿Í $bÀÇ ÇÕ.
$a - $b	»¬¼ÀÀ(Subtraction)	$a¿¡¼ $b¸¦ »« °ª.
$a * $b	°ö¼ÀÀ(Multiplication)	$a¿Í $bÀÇ °ö.
$a / $b	³ª´°¼ÀÀ(Division)	$a¿¡¼ $b¸¦ ³ª´« °ª(È¤ Àº ¸òµ).
$a % $b	³ª¸ÓÁö(Modulus)	$a¿¡¼ $b¸¦ ³ª´« ³ª¸ÓÁö.

³ª´©±â ¿¬»êÀÚ("/") ´Â µÎ °³ÀÇ ÇÇ¿¬»êÀÚ°¡ ¸ðµÎ Á¤¼öºªÀÏ ¶§ (È¤Àº Á¤¼ö·Î °¨È¨µÇ´Â stringÀÏ ¶§), °á°ú°ªÀ¸·Î ³ª´°¼ÀÇ ¸òµ¿ ÇØ´çÇÏ´Â Á¤¼ö¸¦ ¹ÝÈ¯ÇÑ´Ù. µÎ ÇÇ¿¬»êÀÚ°¡ ½Ç¼öÀÏ ¶§´Â ½Ç¼öÀÌ ³ª´°¼ÀÀ» ÇÑ´Ù.

¹®ÀÚ¿- ¿¬»êÀÚ (String Operators)

¹®ÀÚ¿- ¿¬»êÀÚ´Â µü ÇÑ °³¸¸ ÀÖ´Ù. µÎ ¹®ÀÚ¿-À» ¿¬°áÇÏ´Â concatenation ¿¬»êÀÚ (".") ¸¸ÀÌ ÀÖ´Ù.

```
$a = "Hello ";
$b = $a . "World!"; // now $b = "Hello World!"
```

´ëÀÔ ¿¬»êÀÚ (Assignment Operators)

±âº» ´ëÀÔ ¿¬»êÀÚ´Â "="ÀÌ´Ù. ÀÌ ¿¬»êÀÚ¸¦ Ã³À½ °¸¸ÒÀ» ¶§´Â "°°´Ù"¶ó´Â ¿¬»êÀÚ·Î »ý°¢ÇÏ±â ½±Áö¸¸ ÀüÇô ¾Æ´Ï´Ù. ÀÌ°ÍÀÇ ÀÇ¹Ì´Â ¿À¸¥ÂÊÀÇ Ç¥Çö½ÄÀ» °è»êÇÏ¿© ±× °ªÀ» ¿ÞÂÊ¿¡ ÀÖ´Â ÇÇ¿¬»êÚÀÇ °ªÀ¸·Î ¼³Á¤ÇÑ´Ù´Â ÀÇ¹ÌÀÌ´Ù. (ÀÌ°ÍÀ» "gets set to"¶ó°í ÇÑ´Ù.)

´ëÀÔ ¿¬»êÚÀÇ °ªÀº ¿ÞÂÊÀ¸·Î ´ëÀÔµÈ °ªÀÌ´Ù. Áï, "$a = 3"ÀÇ °ªÀº 3ÀÌ µÈ´Ù. ÀÌ°ÍÀº ¿©·¯°Ð¿¡°Ô ¾à°£ÀÇ Æ®¸¯ÀÌ °¡´ÉÇÏµµ ·ÏÇÑ´Ù.

```
$a = ($b = 4) + 5; // $a´Â 9, $b´Â 4ÀÇ °ªÀ» °¡Áø´Ù.
```

±âº»ÀûÀÎ ´ëÀÔ ¿¬»êÀÚ ¿Ü¿¡ ¸ðµç Bit ´ÜÀ§ ¿¬»êÀÚ, »ê¼ú ¿¬»êÀÚ¿Í °áÇÕÇÑ °¹ÇÕ ´ëÀÔ ¿¬»êÚµµ ÀÖ´Ù. °áÇÕµÈ ¿¬»êÀÚ´Â Ç¥Çö½Ä¿¡¼ ÇØ´ç ¿¬»êÀ¸·Î »ç¿ëµÇ°í, ±× ¿¬»ê °ªÀ» ¿ÞÂÊ¿¡ ÀÖ´Â ÇÇ¿¬»êÀÚ¿¡ ´ëÀÔÇÑ´Ù. ¿¹¸¦ µé¾î :

```
$a = 3;
$a += 5; // $a´Â 8ÀÌ´Ù. $a = $a + 5; ¿Í µ¿ÀÏÇÏ´Ù.
$b = "Hello ";
$b .= "There!"; // $b´Â "Hello There!"°¡ µÈ´Ù. $b = $b . "There!";¿Í °°´Ù.
```

Bit ´ÜÀ§ ¿¬»êÀÚ (Bitwise Operators)

°ñÆ® ´ÜÀ§ ¿¬»êÀÚ´Â Á¤¼öÀÇ °³°° °ñÆ®ÀÇ °ªÀ» on, off ÇÒ ¼ö ÀÖ´Ù.

Table 7 - 2. Bitwise Operators

example	name	result
$a & $b	And	$a¿Í $b°¡ ¸ðµÎ 1ÀÎ °ñÆ®¸¸ 1ÀÌ µÈ´Ù.
$a | $b	Or	$a¿Í $bÁß ÇÏ³ª¶óµµ 1ÀÎ °ñÆ®´Â 1ÀÌ µÈ´Ù.
~ $a	Not	$a¿¡¼ 1ÀÎ °ñÆ®´Â 0À¸·Î, 0ÀÎ °ñÆ®´Â 1·Î µÈ´Ù.

Logical Operators

Table 7 - 3. Logical Operators

example	name	result
$a and $b	And	$a¿Í $b°¡ ¸ðµÎ true ÀÌ¸é true
$a or $b	Or	$a¿Í $bÁß ÇÏ³ª¶óµµ true¿ÀÌ¸é true
$a xor $b	Xor	$a¿Í $bÁß ÇÏ³ª¸¸ true¿ÀÌ¸î¸ß true
! $a	Not	$a°¡ true°¡ ¾Æ´Ï¸é true
$a && $b	And	$a¿Í $b°¡ ¸ðµÎ true ÀÌ¸é true
$a || $b	Or	$a¿Í $bÁß ÇÏ³ª¶óµµ true¿ÀÌ¸é true

"and"¿Í "or" ¿¬»êÀÚ°¡ µÎ°¡ÁöÀÇ ´Ù¸¥ ÇüÅÂ¸¦ °¡Áö°í ÀÖ´Â ÀÌÀ¯´Â ÀÌ ¿¬»êÀÚµéÀÌ ¼·Î ´Ù¸¥ ¿ì¼±¼øÀ§¸¦ °¡Áö°í ÀÖ±â ¶§¹®ÀÌ´Ù. (¾Æ·¡¸¦ °¸¶ó.)

°ñ±³ ¿¬»êÀÚ (Comparison Operators)

°ñ±³ ¿¬»êÀÚ´Â ÀÌ¸§¿¡¼ ¾Ë ¼ö ÀÖµíÀÌ µÎ °³ÀÇ °ªÀ» °ñ±³ÇÏ´Â ¿¬»êÀÚÀÌ´Ù.

Table 7 - 4. Comparson Operators

example	name	result
$a = = $b	Equal	$a¿Í $b°¡ °°À¸¸é true.
$a != $b	Not equal	$a¿Í $b°¡ ´Ù¸£¸é true.
$a < $b	Less than	$a°¡ $b°¸´Ù ÀÛÀ¸¸é true.
$a > $b	Greater than	$a°¡ $b°¸´Ù Å©¸é true.
$a <= $b	Less than or equal to	$a°¡ $b°¸´Ù ÀÛ°Å³ª °°À¸¸é true.
$a >= $b	Greater than or equal to	$a°¡ $b°¸´Ù Å©°Å³ª °°À¸¸é true.

¿¬»êÀÚ ¿ì¼± ¼øÀ§ (Precedence)

II. Function Reference

(¿ªÀÚÁÖ: ÀÌ Reference °Î°ÐÀ° °ñ±³Àû °£´ÜÇÑ ³»¿ëµéÀÌ°í, ³»¿ëÀÇ Á¤È®¼º» ±âÇÏ±â À§ÇØ Æ¯°°ÇÑ °æ¿ì¸¦ Á¦¿ÜÇÏ°í´Â Á¦¸ñ °Î°Ð¸¸ ¹ø¿ªÇÏ¿´´Ù.)

Table of Contents

I. Adabas D Functions

Table of Contents

ada_fieldnum
ada_fieldtype
ada_freeresult
ada_numfields
ada_numrows
ada_result
ada_resultall
ada_rollback

Adabas D ÇÔ¼öµéÀº °°·Î ÁÁÀº Æò°¡¸¦ ¹ÞÁö ¸øÇÔ´Ù. ¾Æ¸¶µµ Unified ODBC functionsÀ» ´ë½Å ¾²´Â °ÍÀÌ ÁÁÀ» °ÍÀÌ´Ù.

ada_afetch

ada_afetch - - result row¸¦ ¹è¿·Î °¡Á®¿Â´Ù.

Description

See **odbc_fetch_into()**

ada_autocommit

ada_autocommit - - autocommit µ¿ÀÛÀ» ÇÏ°Å³ª ¾ÈÇÏ°Ô ÇÑ´Ù.

Description

See **odbc_autocommit()**.

ada_close

ada_close - - Adabas D server·ÎÀÇ ¿¬°áÀ» ´Ý´Â´Ù.

Description

See **odbc_close()**.

ada_commit

ada_commit - - transactionÀ» commit ÇÑ´Ù.

Description

See **odbc_commit()**

ada_connect

ada_connect - - Adabas D datasource¿¡ ¿¬°áÇÑ´Ù.

Description

See **odbc_connect()**.

ada_exec

ada_exec - - SQL ¹®ÀåÀ» ÁØºñÇÏ°í ½ÇÇàÇÑ´Ù.

Description

See **odbc_exec()** or **odbc_do()**.

ada_fetchrow

ada_fetchrow - - result·Î ºÎÅÍ row¸¦ °¡Á®¿Â´Ù.

Description

See **odbc_fetch_row()**.

ada_fieldname

ada_fieldname - - columnnameÀ» ±¸ÇÑ´Ù.

Description

See **odbc_field_name()**.

ada_fieldnum

ada_fieldnum - - column number¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_field_num()**.

ada_fieldtype

ada_fieldtype - - fieldÀÇ datatypeÀ» ±¸ÇÑ´Ù.

Description

See **odbc_field_type()**.

ada_freeresult

ada_freeresult - - result¿¡ ÇÒ´çµÈ resource¸¦ ÇØÁ¦ÇÑ´Ù.

Description

See **odbc_free_result()**.

ada_numfields

ada_numfields - - resultÀÇ column °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_num_fields()**.

ada_numrows

ada_numrows - - resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_num_rows()**.

ada_result

ada_result - - result·Î°ÎÅÍ data¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_result()**.

ada_resultall

ada_resultall - - result¸¦ HTML table·Î Ãâ·ÂÇÑ´Ù.

Description

See **odbc_result_all()**.

ada_rollback

ada_rollback - - transactionÀ» rollback ÇÑ´Ù.

Description

See **odbc_rollback()**.

II. Apache Specific Functions

Table of Contents

apache_lookup_uri

apache_lookup_uri - - Æ¯Á¤ URI¿¡ ´ëÇÑ ºÎºÐÀûÀÎ ¿äÃ»(partial request)À» ¼öÇàÇÏ°í ±×°Í¿¡ ´ëÇÑ ¸ðµç Á¤º¸¸¦ µ¹·ÁÁØ´Ù.

Description

`class apache_lookup_uri`(string filename);

This performs a partial request for a URI. It goes just far enough to obtain all the important information about the given resource and returns this information in a class. The properties of the returned class are:
status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

apache_note

apache_note - - apache request note¸¦ ¼³Á¤ÇÏ°Å³ª ÀÐ¾î¿Â´Ù.

Description

`string apache_note`(string note_name, string [note_value]);

apache_note() is an Apache-specific function which gets and sets values in a request's `notes` table. If called with one argument, it returns the current value of note `note_name`. If called with two arguments, it sets the value of note `note_name` to `note_value` and returns the previous value of note `note_name`.

getallheaders

getallheaders - - ¸ðµç HTTP request header¸¦ ºÐ¼®ÇÏ¿© ÀÐ¾î¿Â´Ù.

Description

```
array getallheaders(void);
```

This function returns an associative array of all the HTTP headers in the current request.
Example 1. GetAllHeaders() Example

```
$headers = getallheaders();
while (list($header, $value) = each($headers)) {
    echo "$header: $value<br>\n";
}
```

This example will display all the request headers for the current request.

> **Note:** **GetAllHeaders()** is currently only supported when PHP runs as an Apache module.

virtual

virtual - - Apache sub - request¸¦ ¼öÇàÇÑ´Ù.

Description

```
int virtual(string filename);
```

virtual() is an Apache - specific function which is equivalent to <! - - #include virtual... - - > in mod_include. It performs an Apache sub - request. It is useful for including CGI scripts or .shtml files, or anything else that you would parse through Apache. Note that for a CGI script, the script must generate valid CGI headers. At the minimum that means it must generate a Content - type header. For PHP files, you should use **include()** or **require()**.

III. Array Functions

Table of Contents

array

array - - ¹è¿-À» ¸¸µç´Ù.

Description

array array(...);

Returns an array of the parameters. The parameters can be given an index with the => operator.

Note that **array()** really is a language construct used to represent literal arrays, and not a regular function.

The following example demonstrates how to create a two - dimensional array, how to specify keys for associative arrays, and how to skip - and - continue numeric indices in normal arrays.
Example 1. array() example

```
$fruits = array(
```

```
    "fruits"  => array("a"=>"orange","b"=>"banana","c"=>"apple"),
    "numbers" => array(1, 2, 3, 4, 5, 6)
    "holes"   => array("first", 5 => "second", "third")
);
```

See also: **list()**.

array_walk

array_walk - - ¹è¿-ÀÇ °³°³ÀÇ ¿ø¼Ò¿¡ Æ¯Á¤ ÇÔ¼ö¸¦ Àû¿ëÇÏ¿© ¼öÇàÇÑ´Ù.

Description

int array_walk(array arr, string func);

Applies the function named by *func* to each element of *arr*. The elements are passed as the first argument of *func*; if *func* requires more than one argument, a warning will be generated each time **array_walk()** calls *func*. These warnings may be suppressed by prepending the '@' sign to the **array_walk()** call, or by using **error_reporting()**.

Note that *func* will actually be working with the elements of *arr*, so any changes made to those elements will actually be made in the array itself.
Example 1. array_walk() example

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
function test_alter( $item1 ) {
   $item1 = 'bogus';
}
function test_print( $item2 ) {
   echo "$item2<br>\n";
}
array_walk( $fruits, 'test_print' );
array_walk( $fruits, 'test_alter' );
array_walk( $fruits, 'test_print' );
```

See also **each()** and **list()**.

arsort

arsort - - ¹è¿-À» ¿ª¼øÀ¸·Î Á¤·ÄÇÏ°í index associationÀ» À¯ÁöÇÑ´Ù.

Description

void arsort(array array);

This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant.
Example 1. arsort() example

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
arsort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

This example would display: `fruits[a] = orange fruits[d] = lemon fruits[b] = banana fruits[c] = apple` The fruits have been sorted in reverse alphabetical order, and the index associated with each element has been maintained.

See also: **asort()**, **rsort()**, **ksort()**, and **sort()**.

asort

asort - - ¹è¿-À» Á¤·ÄÇÏ°í index associationÀ» À¯ÁöÇÑ´Ù.

Description

void asort(array array);

This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant.
Example 1. asort() example

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
asort($fruits);
```

```
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

This example would display: `fruits[c] = apple fruits[b] = banana fruits[d] = lemon fruits[a] = orange` The fruits have been sorted in alphabetical order, and the index associated with each element has been maintained.

See also **arsort()**, **rsort()**, **ksort()**, and **sort()**.

count

count - - ¹è¿- º¯¼öÀÇ ¿ø¼Ò °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int count(mixed var);

Returns the number of elements in *var*, which is typically an array (since anything else will have one element).

Returns 0 if the variable is not set.

Returns 1 if the variable is not an array.

See also: **sizeof()**, **isset()**, and **is_array()**.

current

current - - ¹è¿-ÀÇ ÇöÀç ¿ø¼Ò¸¦ µ¹·ÁØ´Ù.

Description

mixed current(array array);

°¢°¢ÀÇ ¹è¿- º¯¼ö´Â ±×°ÍÀÇ ¿ø¼Ò¸¦ °¡¸£Å°´Â ³»ºÎÀûÀÎ pointer¸¦ °¡Áö°í ÀÖ´Ù. °Ô´Ù°¡, ¹è¿-ÀÇ ¸ðµç ¿ø¼ÒµéÀº °Ë»öÀÌ ¿ëÀÌÇÏµµ·Ï ¾ç¹æÇâ linked list·Î ¿¬°áµÇ¾î ÀÖ´Ù. ÀÌ ³»ºÎÀûÀÎ pointer´Â ´Ù¸¥ ¾î¶² Á¶ÀÛ»ó ÇÏ±â Àü¿¡´Â Ç×»ó Ã¹ ¹øÂ° ¿ä¼Ò¸¦ °¡¸£Å°°í ÀÖ´Ù.

current() ÇÔ¼ö´Â ´Ü¼øÈ÷ ³»ºÎÀûÀÎ Pointer°¡ °¡¸£Å°°í ÀÖ´Â ¿ø¼Ò¸¦ ¹ÝÈ¯ÇÒ »ÓÀÌ´Ù. ¸¸¾à ÀÌ pointer°¡ ¿ø¼Ò listÀÇ ¹üÀ§¸¦ ³Ñ¾î¼- ÁöÀÖÇÏ°í ÀÖ´Ù¸é **current()**´Â false¸¦ ¹ÝÈ¯ÇÑ´Ù.

(¿ªÀÚÁÖ : ¹ö±×ÀÎ°Í °°Áö¸¸, **current()**´Â ÇöÀç ¿ø¼Ò°¡ 0ÀÌ³ª ""(°ó ¹®ÀÚ¿-)ÀÇ °ªÀ» °¡Áö°í ÀÖÀ¸¸é falseÀ» ¹ÝÈ¯ÇÑ´Ù. µû¶ó¼-ÀÌ **current()** ÇÔ¼ö´Â ¿ø¼ÒÀÇ °ªÀÌ 0ÀÎ°¡ ¾Æ´Ï¸é ¹è¿-ÀÇ ¹üÀ§¸¦ ³Ñ¾ú´Â°¡¸¦ ÆÇ´ÜÇÒ ¼ö ¾ø´Ù. **current()**¸¦ »ç¿ëÇÑ loop ÄÚµù°¸´Ù´Â **each()** ÇÔ¼ö¸¦ »ç¿ëÇÏ´Â °ÍÀÌ ÁÁ´Ù.)

See also: **end()**, **next()**, **prev()** and **reset()**.

each

each - - ¹è¿-¿¡¼- ´ÙÀ½ key/value ½ÖÀ» µ¹·ÁØ´Ù.

Description

array each(array array);

array ¹è¿-¿¡¼- ´ÙÀ½ key/value ½ÖÀ» ¹ÝÈ¯ÇÑ´Ù. ÀÌ ½ÖÀº ³× °³ÀÇ ¿ø¼Ò¸¦ °¡Áø ¹è¿-·Î ¹ÝÈ¯µÇ´Â¥ ÀÌ ³× °³ÀÇ ¿ø¼ÒÀÇ key´Â *0*, *1*, *key*, *value*ÀÌ´Ù. *0*°ú *key* ¿ø¼Ò´Â °¢°¢ º¯¼öÀÇ key ÀÌ¸§À» °¡Áö°í, *1*°ú *value*´Â ±× °ªÀ» °¡Áö°í ÀÖ´Ù.
Example 1. each() examples

```
$foo = array( "bob", "fred", "jussi", "jouni" );
$bar = each( $foo );
```

`$bar` now contains the following key/value pairs:

```
0 => 0
1 => 'bob'
key => 0
value => 'bob'
```

```
$foo = array( "Robert", => "Bob", "Seppo" => "Sepi" );
```

```
$bar = each( $foo );
```

$bar now contains the following key/value pairs:

```
    0 => 'Robert'
    1 => 'Bob'
    key => 'Robert'
    value => 'Bob'
```

º¸Åë **each()**´Â **list()** ÇÔ¼öº¿Í ÇÔ²² ¹è¿-À» Å½»öÇÏ´Âµ¥ »ç¿ëµÈ´Ù. ¿¹¸¦ µé¾î ´ÙÀ½°ú °°ÀÌ $HTTP_POST_VARS¿Í °°Àº ¹è¿-À» Å½»öÇÏ´Âµ¥ ¸¦ÀÌ »ç¿ëµÈ´Ù

Example 2. Traversing $HTTP_POST_VARS with each()

```
echo "Values submitted via POST method:<br>";
while ( list( $key, $val ) = each( $HTTP_POST_VARS ) ) {
   echo "$key => $val<br>";
}
```

See also **key()**, **current()**, **reset()**, **next()**, and **prev()**.

end

end - - ¹è¿-ÀÇ ³»ºÎÀûÀÎ(internal) pointer¸¦ ¸Ç ¸¶Áö¸· ¿ø¼Ò·Î ¿Å±ä´Ù.

Description

end(array array);

end() advances *array*'s internal pointer to the last element.

See also: **current()**, **end()** **next()** and **reset()**

key

key - - ¹è¿-(associative array)¿¡¼- ³»ºÎÀûÀÎ(internal) pointer°¡ °¡¸®Å°°í ÀÖ´Â ¿ø¼ÒÀÇ key °ªÀ» °¡Á®¿Â´Ù.

Description

mixed prev(array array);

key() returns the index element of the current array position.

See also: **current()** **next()**

ksort

ksort - - key ¼øÀ¸·Î ¹è¿-À» Á¤·ÄÇÑ´Ù.

Description

int ksort(array array);

Sorts an array by key, maintaining key to data correlations. This is useful mainly for associative arrays.

Example 1. ksort() example

```
$fruits = array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
ksort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

This example would display: `fruits[a] = orange fruits[b] = banana fruits[c] = apple fruits[d] = lemon`

See also **asort()**, **arsort()**, **sort()**, and **rsort()**.

list

list - - º¯¼öº¸¸¦ ¹è¿-ÀÎ °ÍÃ³·³ ¸¸µç´Ù.

Description

void list(...);

Like **array()**, this is not really a function, but a language construct. **list()** is used to assign a list of variables in one operation.

Example 1. list() example

```
<table>
 <tr>
  <th>Employee name</th>
  <th>Salary</th>
 </tr>
<?php
$result = mysql($conn, "SELECT id, name, salary FROM employees");
while (list($id, $name, $salary) = mysql_fetch_row($result)) {
    print(" <tr>\n".
          "  <td><a href=\"info.php3?id=$id\">$name</a></td>\n".
          "  <td>$salary</td>\n".
          " </tr>\n");
}
?></table>
```

See also: **array()**.

next

next - - ¹è¿-ÀÇ internal pointer¸¦ ÇÏ³ª ÀüÁø½ÃÅ²´Ù.

Description

mixed next(array array);

Returns the array element in the next place that's pointed by the internal array pointer, or false if there are no more elements.

next() behaves like **current()**, with one difference. It advances the internal array pointer one place forward before returning the element. That means it returns the next array element and advances the internal array pointer by one. If advancing the internal array pointer results in going beyond the end of the element list, **next()** returns false.

See also: **current()**, **end() prev()** and **reset()**

pos

pos - - ¹è¿-ÀÇ ÇöÀç ¿ø¼Ò¸¦ returnÇÑ´Ù.

Description

mixed pos(array array);

This is an alias for **current()**.

See also: **end()**, **next()**, **prev()** and **reset()**.

prev

prev - - ¹è¿-ÀÇ internal pointer¸¦ ÇÏ³ª µÚ·Î ÈÄÁø½ÃÅ²´Ù.

Description

mixed prev(array array);

Returns the array element in the previous place that's pointed by the internal array pointer, or false if there are no more elements.

prev() behaves just like **next()**, except it rewinds the internal array pointer one place instead of advancing it.

See also: **current()**, **end() next()** and **reset()**

reset

reset - - ¹è¿-ÀÇ internal pointer¸¦ ¸Ç Ã³À½ ¿ø¼Ò·Î ¼³Á¤ÇÑ´Ù.

Description

reset(array array);

reset() rewinds *array*'s internal pointer to the first element.

See also: **current()**, **next() prev()** and **reset()**

rsort

rsort - - ¹è¿-À» ¿ª¼øÀ¸·Î Á¤·ÄÇÑ´Ù.

Description

void rsort(array array);

This function sorts an array in reverse order (highest to lowest).
Example 1. rsort() example

```
$fruits = array("lemon","orange","banana","apple");
rsort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

This example would display: `fruits[0] = orange fruits[1] = lemon fruits[2] = banana fruits[3] = apple` The fruits have been sorted in reverse alphabetical order.

See also **arsort()**, **asort()**, **ksort()**, and **sort()**.

sizeof

sizeof - - ¹è¿-ÀÇ Å©±â¸¦ ±¸Ñ´Ù. ¿ø¼ÒÀÇ °³¼ö¸¦ ±¸ÇØÁø´Ù.

Description

int sizeof(array array);

Returns the number of elements in the array.

See also: **count()**

sort

sort - - ¹è¿-À» Á¤·ÄÇÑ´Ù.

Description

void sort(array array);

This function sorts an array. Elements will be arranged from lowest to highest when this function has completed.
Example 1. sort() example

```
$fruits = array("lemon","orange","banana","apple");
sort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

This example would display: `fruits[0] = apple fruits[1] = banana fruits[2] = lemon fruits[3] = orange` The fruits have been sorted in alphabetical order.

See also **arsort()**, **asort()**, **ksort()**, **rsort()**, and **usort()**.

uasort

uasort - - »ç¿ëÀÚ°¡ ÁöÁ¤ÇÑ ºñ±³ ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© Á¤·ÄÇÏ°í index associationÀ» À¯ÁöÇÑ´Ù.

Description

void uasort(array array, function cmp_function);

This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant. The comparison function is user - defined.

uksort

uksort - - »ç¿ëÀÚ°¡ ÁöÁ¤ÇÑ °ñ±³ ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© key ¼øÀ¸·Î Á¤·ÄÇÑ´Ù.

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

Description

void uksort(array array, function cmp_function);

This function will sort the keys of an array using a user - supplied comparison function. If the array you wish to sort needs to be sorted by some non - trivial criteria, you should use this function.
Example 1. uksort() example

```
function mycompare($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array(4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");
uksort($a, mycompare);
while(list($key, $value) = each($a)) {
    echo "$key: $value\n";
}
```

This example would display: 20: twenty 10: ten 4: four 3: three

See also **arsort()**, **asort()**, **uasort()**, **ksort()**, **rsort()** and **sort()**.

usort

usort - - »ç¿ëÀÚ°¡ ÁöÁ¤ÇÑ °ñ±³ ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© value ¼øÀ¸·Î Á¤·ÄÇÑ´Ù.

Description

void usort(array array, function cmp_function);

This function will sort an array by its values using a user - supplied comparison function. If the array you wish to sort needs to be sorted by some non - trivial criteria, you should use this function.
Example 1. usort() example

```
function cmp($a,$b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array(3,2,5,6,1);
usort($a, cmp);
while(list($key,$value) = each($a)) {
    echo "$key: $value\n";
}
```

This example would display: 0: 6 1: 5 2: 3 3: 2 4: 1 Obviously in this trivial case the **rsort()** function would be more appropriate.

See also **arsort()**, **asort()**, **ksort()**, **rsort()** and **sort()**.

IV. Aspell Functions

Table of Contents

aspell() ÇÔ¼öµéÀº ¿©·¯°Ð¿¡°Ô ´Ü¾îÀÇ Ã¶ÀÚ¸¦ °Ë»çÇÏ°í ¼öÁ¤»çÇ×À» Á¶¾ðÇØ ÁØ´Ù.

ÀÌ ÇÔ¼öµéÀ» »ç¿ëÇÏ·Á¸é aspell ¶óÀÌºê·¯¸®¸¦ ÇÊ¿äÇÏ´Ù. ´ÙÀ½ »çÀÌÆ®¿¡¼ Ã£À» ¼ö ÀÖÀ» °ÍÀÌ´Ù. : http://metalab.unc.edu/kevina/aspell/

aspell_new

aspell_new - - »õ »çÀüÀ» ÀÐ¾î ¿Â´Ù.

Description

int aspell_new(string master, string personal);

aspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other aspell functions.
Example 1. aspell_new

```
$aspell_link=aspell_new("english");
```

aspell_check

aspell_check - - ´Ü¾î¸¦ °Ë»çÇÑ´Ù.

Description

boolean aspell_check(int dictionary_link, string word);

aspell_check() checks the spelling of a word and returns true if the spelling is correct, false if not.
Example 1. aspell_check

```
$aspell_link=aspell_new("english");
if (aspell_check($aspell_link,"testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_check-raw

aspell_check-raw - - ´Ü¾îÀÇ ´ë/¼Ò¹®ÀÚ¸¦ °¨¾È½ÃÅ°Áö ¾Ê°í, ¾ÕµÚÀÇ °ø¹éµµ Á¦°ÅÇÏÁö ¾ÊÀº Ã¤·Î °Ë»çÇÑ´Ù.

Description

boolean aspell_check_raw(int dictionary_link, string word);

aspell_check_raw() checks the spelling of a word, without changing its case or trying to trim it in any way and returns true if the spelling is correct, false if not.
Example 1. aspell_check_raw

```
$aspell_link=aspell_new("english");
if (aspell_check_raw($aspell_link,"testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_suggest

aspell_suggest - - ´Ü¾îÀÇ Ã¶ÀÚ¸¦ Á¶¾ðÇØ ÁØ´Ù.

Description

array aspell_suggest(int dictionary_link, string word);

aspell_suggest() returns an array of possible spellings for the given word.
Example 1. aspell_suggest

```
$aspell_link=aspell_new("english");
if (!aspell_check($aspell_link,"testt")) {
    $suggestions=aspell_suggest($aspell_link,"testt");
    for($i=0; $i < count($suggestions); $i++) {
       echo "Possible spelling: " . $suggestions[$i] . "<br>";
```

```
    }
}
```

V. BC (Arbitrary Precision) Functions

Table of Contents

ÀÌ BC ÇÔ¼öµéÀº PHP°¡ --enable-bcmath ¼³Á¤ ¿É¼ÇÀ¸·Î ÄÄÆÄÀÏ µÈ °æ¿ì¿¡¸¸ »ç¿ëÀÌ °¡´ÉÇÏ´Ù.

bcadd

bcadd -- µÎ °³ÀÇ arbitrary precision number¸¦ ´õÇÑ´Ù.

Description

string bcadd(string left operand, string right operand, int [scale]);

Adds the *left operand* to the *right operand* and returns the sum in a string. The optional *scale* parameter is used to set the number of digits after the decimal place in the result.

See also **bcsub()**.

bccomp

bccomp -- µÎ arbitrary precision numbers¸¦ ºñ±³ÇÑ´Ù.

Description

int bccomp(string left operand, string right operand, int [scale]);

Compares the *left operand* to the *right operand* and returns the result as an integer. The optional *scale* parameter is used to set the number of digits after the decimal place which will be used in the comparion. The return value is 0 if the two operands are equal. If the *left operand* is larger than the *right operand* the return value is +1 and if the *left operand* is less than the *right operand* the return value is -1.

bcdiv

bcdiv -- µÎ arbitrary precision number¸¦ ³ª´«´Ù.

Description

string bcdiv(string left operand, string right operand, int [scale]);

Divides the *left operand* by the *right operand* and returns the result. The optional *scale* sets the number of digits after the decimal place in the result.

See also **bcmul()**.

bcmod

bcmod -- arbitrary precision numberÀÇ ³ª¸ÓÁö¸¦ ±¸ÇÑ´Ù.

Description

string bcmod(string left operand, string modulus);

Get the modulus of the *left operand* using *modulus*.

See also **bcdiv()**.

bcmul

bcmul - - µÎ arbitrary precision number¸¦ °öÇÑ´Ù.

Description

string bcmul(string left operand, string right operand, int [scale]);

Multiply the *left operand* by the *right operand* and returns the result. The optional *scale* sets the number of digits after the decimal place in the result.

See also **bcdiv()**.

bcpow

bcpow - - arbitrary precision numberÀÇ n Á¦°öÇÑ´Ù.

Description

string bcpow(string x, string y, int [scale]);

Raise *x* to the power *y*. The *scale* can be used to set the number of digits after the decimal place in the result.

See also **bcsqrt()**.

bcscale

bcscale - - ¸ðµç bc ¼öÇÐ ÇÔ¼öÀÇ ±â°» scale parameter¸¦ Á¤ÇÑ´Ù.

Description

string bcscale(int scale);

This function sets the default scale parameter for all subsequent bc math functions that do not explicitly specify a scale parameter.

bcsqrt

bcsqrt - - arbitray precision numberÀÇ Á¦°ö±ÙÀ» ¸¸µçÑ´Ù.

Description

string bcsqrt(string operand, int scale);

Return the square root of the *operand*. The optional *scale* parameter sets the number of digits after the decimal place in the result.

See also **bcpow()**.

bcsub

bcsub - - arbitrary precision number ¸¦ »«´Ù.

Description

string bcsub(string left operand, string right operand, int [scale]);

Subtracts the *right operand* from the *left operand* and returns the result in a string. The optional *scale* parameter is used to set the number of digits after the decimal place in the result.

See also **bcadd()**.

VI. Calendar Functions

Table of Contents

PHP¿¡´Â ¼-·Î ´Ù¸¥ ³¯Â¥(´Þ·Â) ÇüÅÂ¸¦ °¨È¯½ÃÄÑ ÁÖ´Â ÇÔ¼öµéÀ» Á¦°øÇÑ´Ù. Julian Day Count°¡ ±â°»ÀÌ µÈ´Ù. ÀÌ°ÍÀº BC4000³âÀÇ ¼îÀ´ ½ÃÁ¡À» ±âÁØÀ¸·Î ÀâÀ¾Æ ±×°÷¿¡¼-°ÎÅÍ ¾ó¸¶ÀÇ ³¯Â¥°¡ Áö³µ´ÂÁö¸¦ ±âÁØÀ¸·Î »ï´Â °ÍÀÌ´Ù. ÀÌ Julian Day Count´Â ÀÏ¹ÝÀûÀ¸·Î »ç¿ëÇÏ´Â Julian ´Þ·Â°ú´Â ´Ù¸£´Ù´Â °ÍÀ» ¾Ë¾Æµ¶ÎÀÚ. Calendar ½Ã½º°ÅÛ¿¡ ´ëÇØ ÀÚ¼¼È÷ ¾Ë°í ½Í´Ù¸é http://genealogy.org/~scottlee/cal-overview.html¸¦ ¹æ¹®ÇÏ¿© °¸¶ó. ÀÌ ¼³¸í¼-Áß¿¡¼-´Â À§ÀÇ ÆäÀÌÁö¿¡¼- ¹ßÃéµÈ ³»¿ë À» ""·Î µÑ·¯ ½Î°í ÀÖ´Ù.

(¿ªÀÚÁÖ : ÀÌ ÇÔ¼öµéÀº dl/calender extensionÀ» LoadÇÑ ÈÄ¿¡ »ç¿ë°¡´ÉÇÏ´Ù. dl/README ÆÄÀÏÀ» ÀÐ¾î°¸¶ó.)

JDToGregorian

JDToGregorian - - Julian Day Count¸¦ Gregorian date·Î °¯È¯

Description

string jdtogregorian(int julianday);

Converts Julian Day Count to a string containing the Gregorian date in the format of "month/day/year"

GregorianToJD

GregorianToJD - - Gregorian date¸¦ Julian Day Count·Î °¯È¯

Description

int gregoriantojd(int month, int day, int year);

Valid Range for Gregorian Calendar 4714 B.C. to 9999 A.D.

Although this software can handle dates all the way back to 4714 B.C., such use may not be meaningful. The Gregorian calendar was not instituted until October 15, 1582 (or October 5, 1582 in the Julian calendar). Some countries did not accept it until much later. For example, Britain converted in 1752, The USSR in 1918 and Greece in 1923. Most European countries used the Julian calendar prior to the Gregorian.

Example 1. Calendar functions

```
<?php
$jd = GregorianToJD(10,11,1970);
echo("$jd\n");
$gregorian = JDToGregorian($jd);
echo("$gregorian\n");
?>
```

JDToJulian

JDToJulian - - Julian Day Count¸¦ Julian Calendar date·Î °¯È¯

Description

string jdtojulian(int julianday);

Converts Julian Day Count to a string containing the Julian Calendar Date in the format of "month/day/year".

JulianToJD

JulianToJD - - Julian Calendar date¸¦ Julian Day Count·Î °¯È¯

Description

int juliantojd(int month, int day, int year);

Valid Range for Julian Calendar 4713 B.C. to 9999 A.D.

Although this software can handle dates all the way back to 4713 B.C., such use may not be meaningful. The calendar was created in 46 B.C., but the details did not stabilize until at least 8 A.D., and perhaps as late at the 4th century. Also, the beginning of a year varied from one culture to another - not all accepted January as the first month.

JDToJewish

JDToJewish - - Julian Day Count¸¦ the À¯´ë Calendar·Î º¯È¯.

Description

string jdtojewish(int julianday);

Converts a Julian Day Count the the Jewish Calendar.

JewishToJD

JewishToJD - - À¯´ë Calendar¸¦ Julian Day Count·Î º¯È¯.

Description

int jewishtojd(int month, int day, int year);

Valid Range Although this software can handle dates all the way back to the year 1 (3761 B.C.), such use may not be meaningful.

The Jewish calendar has been in use for several thousand years, but in the early days there was no formula to determine the start of a month. A new month was started when the new moon was first observed.

JDToFrench

JDToFrench - - Julian Day Count¸¦ French Republican Calendar·Î º¯È¯.

Description

string jdtofrench(int month, int day, int year);

Converts a Julian Day Count to the French Republican Calendar.

FrenchToJD

FrenchToJD - - French Republican Calendar¸¦ Julian Day Count·Î º¯È¯.

Description

int frenchtojd(int month, int day, int year);

Converts a date from the French Republican Calendar to a Julian Day Count

These routines only convert dates in years 1 through 14 (Gregorian dates 22 September 1792 through 22 September 1806). This more than covers the period when the calendar was in use.

JDMonthName

JDMonthName - - ¿ùÀÇ ÀÌ¸§À» ¹ÝÈ¯

Description

string jdmonthname(int julianday, int mode);

Returns a string containing a month name. *mode* tells this function which calendar to convert the Julian Day Count

to, and what type of month names are to be returned.

Table 1. Calendar modes

Mode	Meaning
0	Gregorian - apreviated
1	Gregorian
2	Julian - apreviated
3	Julian
4	Jewish
5	French Republican

JDDayOfWeek

JDDayOfWeek - - ÇØ´ç ³¯Â¥ÀÇ ¿äÀÏÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

mixed jddayofweek(int julianday, int mode);

Returns the day of the week. Can return a string or an int depending on the mode.

Table 1. Calendar week modes

Mode	Meaning
0	returns the day number as an int (0=sunday, 1=monday, etc)
1	returns string containing the day of week (english - gregorian)
2	returns a string containing the abreviated day of week (english - gregorian)

VII. Date/Time Functions

Table of Contents

checkdate

checkdate - - date/time°ªÀÌ ¿Ã¹Ù¸¥°¡ °Ë»çÇÑ´Ù.

Description

int checkdate(int month, int day, int year);

Returns true if the date given is valid; otherwise returns false. Checks the validity of the date formed by the arguments. A date is considered valid if:

year is between 1900 and 32767 inclusive

month is between 1 and 12 inclusive

day is within the allowed number of days for the given month. Leap years are taken into consideration.

date

date - - local timeÀ» ÁöÁ¤µÈ ÇüÅÂ·Î ¸¸¾î ¹ÝÈ¯ÇÑ´Ù.

Description

string date(string format, int timestamp);

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given.

The following characters are recognized in the format string:

a - "am" or "pm"
A - "AM" or "PM"
d - day of the month, numeric, 2 digits (with leading zeros)
D - day of the week, textual, 3 letters; i.e. "Fri"
F - month, textual, long; i.e. "January"
h - hour, numeric, 12 hour format
H - hour, numeric, 24 hour format
i - minutes, numeric
j - day of the month, numeric, without leading zeros
l (lowercase 'L') - day of the week, textual, long; i.e. "Friday"
m - month, numeric
M - month, textual, 3 letters; i.e. "Jan"
s - seconds, numeric
S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd"
U - seconds since the epoch
Y - year, numeric, 4 digits
w - day of the week, numeric, 0 represents Sunday
y - year, numeric, 2 digits
z - day of the year, numeric; i.e. "299"

Unrecognized characters in the format string will be printed as-is.
Example 1. date() example

```
print(date( "l dS of F Y h:i:s A" ));
print("July 1, 2000 is on a " . date("l", mktime(0,0,0,7,1,2000)));
```

It is possible to use **date()** and **mktime()** together to find dates in the future or the past.
Example 2. date() and **mktime()** example

```
$tomorrow  = mktime(0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime(0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime(0,0,0,date("m"),  date("d",   date("Y")+1);
```

To format dates in other languages, you should use the **setlocale()** and **strftime()** functions.

See also **gmdate()** and **mktime()**.

strftime

strftime - - ÇöÀç ½Ã°£/³¯Â¥¸¦ ÁöÁ¤ÇÑ ÇüÅÂ·Î ³¯Â¥°ú Ç¥Ù.

Description

string strftime(string format, int timestamp);

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given. Month and weekday names and other language dependent strings respect the current locale set with **setlocale()**.

The following conversion specifiers are recognized in the format string:

%a - abbreviated weekday name according to the current locale
%A - full weekday name according to the current locale
%b - abbreviated month name according to the current locale
%B - full month name according to the current locale
%c - preferred date and time representation for the current locale
%d - day of the month as a decimal number (range 0 to 31)
%H - hour as a decimal number using a 24-hour clock (range 00 to 23)
%I - hour as a decimal number using a 12-hour clock (range 01 to 12)
%j - day of the year as a decimal number (range 001 to 366)
%m - month as a decimal number (range 1 to 12)
%M - minute as a decimal number
%p - either `am' or `pm' according to the given time value, or the corresponding strings for the current locale

%S - second as a decimal number
%U - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week
%W - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week
%w - day of the week as a decimal, Sunday being 0
%x - preferred date representation for the current locale without the time
%X - preferred time representation for the current locale without the date
%y - year as a decimal number without a century (range 00 to 99)
%Y - year as a decimal number including the century
%Z - time zone or name or abbreviation
%% - a literal `%' character

Example 1. strftime() example

```
setlocale ("LC_TIME", "C");
print(strftime("%A in Finnish is "));
setlocale ("LC_TIME", "fi");
print(strftime("%A, in French "));
setlocale ("LC_TIME", "fr");
print(strftime("%A and in German "));
setlocale ("LC_TIME", "de");
print(strftime("%A.\n"));
```

This example works if you have the respective locales installed in your system.

See also **setlocale()** and **mktime()**.

getdate

getdate - - date/time Á¤°¸¸¦ ¾ò´Â´Ù.

Description

array getdate(int timestamp);

Returns an associative array containing the date information of the timestamp as the following array elements:

"seconds" - seconds
"minutes" - minutes
"hours" - hours
"mday" - day of the month
"wday" - day of the week, numeric
"mon" - month, numeric
"year" - year, numeric
"yday" - day of the year, numeric; i.e. "299"
"weekday" - day of the week, textual, full; i.e. "Friday"
"month" - month, textual, full; i.e. "January"

gettimeofday

gettimeofday - - ÇöÀç ½Ã°¢À» ¾ò´Â´Ù.

Description

`array gettimeofday`(void);

This is an interface to gettimeofday(2). It returns an associative array containing the data returned from the system call.

"sec" - seconds
"usec" - microseconds
"minuteswest" - minutes west of Greenwich
"dsttime" - type of dst correction

gmdate

gmdate - - GMT/CUT date/timeÀ» ÁöÁ¤µÈ ÇüÅÂ·Î ¸¸µé¾î ¹ÝÈ¯ÇÑ´Ù.

Description

string gmdate(string format, int timestamp);

Identical to the **date()** function except that the time returned is Greenwich Mean Time (GMT). For example, when run in Finland (GMT +0200), the first line below prints "Jan 01 1998 00:00:00", while the second prints "Dec 31 1997 22:00:00".
Example 1. gmdate() example

```
echo date( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
echo gmdate( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
```

See also **date()**, **mktime()** and **gmmktime()**.

mktime

mktime - - ÁöÁ¤µÈ ³¯Â¥¸¦ date()ÇÔ¼ö µîÀÌ »ç¿ëÇÏ´Â timestamp·Î ¸¸µç´Ù.

Description

int mktime(int hour, int minute, int second, int month, int day, int year);

Returns the Unix timestamp corresponding to the arguments given. This timestamp is a long integer containing the number of seconds between the Unix Epoch (January 1 1970) and the time specified.

Arguments may be left out in order from right to left; any arguments thus omitted will be set to the current value according to the local date and time.

MkTime is useful for doing date arithmetic and validation, as it will automatically calculate the correct value for out-of-range input. For example, each of the following lines produces the string "Jan-01-1998".
Example 1. mktime() example

```
echo date( "M-d-Y", mktime(0,0,0,12,32,1997) );
echo date( "M-d-Y", mktime(0,0,0,13,1,1997) );
echo date( "M-d-Y", mktime(0,0,0,1,1,1998) );
```

(¿ªÀÚÁÖ : ´ÙÀ½°ú °°ÀÌ date() ÇÔ¼ö¸¦ °°ÀÌ »ç¿ëÇÏ¸é °ú°Å³ª ¹Ì·¡ÀÇ Æ¯Á¤ÀÏÀÌ» ¾Ë ¼ö ÀÖ´Ù.
$tomorrow = mktime(0,0,0,date("d")+1,date("m"), date("Y"));
$lastmonth = mktime(0,0,0,date("d"),date("m")-1,date("Y"));
$nextyear = mktime(0,0,0,date("d"),date("m"),date("Y")+1);)

See also **date()** and **time()**.

gmmktime

gmmktime - - GMT ³¯Â¥¸¦ °¡Áö°í timestamp¸¦ ¸¸µç´Ù.

Description

int gmmktime(int hour, int minute, int second, int month, int day, int year);

Identical to **mktime()** except the passed parameters represents a GMT date.

time

time - - ÇöÀç ½Ã°¢ÀÇ timestamp¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int time(void);

Returns the current time measured in the number of seconds since the Unix Epoch (January 1, 1970).

See also **date()**.

microtime

microtime - - ÇöÀç ½Ã°¢ÀÇ timestamp¸¦ 1000°ÐÀÇ 1ÃÊ ´ÜÀ§±îÁö ¹ÝÈ¯ÇÑ´Ù.

Description

string microtime(void);

Returns the string "msec sec" where sec is the current time measured in the number of seconds since the Unix Epoch (0:00:00 January 1, 1970 GMT), and msec is the microseconds part. This function is only available on operating systems that support the gettimeofday() system call.

See also **time()**.

VIII. DBA functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº Berkeley DB ÇüÅÂÀÇ µ¥ÀÌÅÍº£ÀÌ½º¸¦ Á¢±ÙÇÏ±â À§ÇÑ °ÍÀÌ´Ù.

ÀÌ°ÍÀº ¿©·¯ file-based µ¥ÀÌÅÍº£ÀÌ½ºÀÇ ÀÏ¹ÝÀûÀÎ abstraction layerÀÌ´Ù. ±â´ÉÀûÀ¸·Î ÀÌ°ÍÀº Sleepycat Software's DB2 °°Àº Çö´ëÀûÀÎ µ¥ÀÌÅÍº£ÀÌ½º ±â´ÉÀÇ °Î°Ð ÁýÇÕ(subset)ÀÌ µÈ´Ù.

±âÃÊ°¡ µÇ´Â µ¥ÀÌÅÍº£ÀÌ½ºÀÇ ±¸Çö¿¡ µû¶ó ¾ó¸¶³ª ´Ù¾çÇÏ°Ô »ç¿ëµÉ ¼ö ÀÖ´Â°¡ °áÁ¤µÈ´Ù. **dba_optimize()**³ª **dba_sync()** °°Àº ÇÔ¼öó´Â ´ÜÁö ÇÑ µ¥ÀÌÅÍº£ÀÌ½º¿¡ ´ëÇØ ÀÛµ¿ÇÏ´Â °Í¸¸À» ¿µÇâÀ» ¹ÌÄ¡Áö ¸øÇÑ´Ù.

´ÙÀ½°ú °°Àº handler¸¦ Áö¿øÇÑ´Ù. :

dbmÀº °¡Àå ¿À·¡µÈ(original) ÇüÅÂÀÇ Berkeley DB style µ¥ÀÌÅÍº£ÀÌ½ºÀÌ´Ù. °¡´ÉÇÏ¸é ÀÌ°ÍÀº »ç¿ëÏÁö ¾Ê´Â °ÍÀÌ ÁÁ´Ù. ¿ì¸®´Â DB2¿¡ »ç¿ëµÈ ÇÔ¼öµé°úÀÇ È£È¯¼ºÀ» Á¦°øÇÏÁö ¾Ê´Â´Ù. ¿Ö³ÅÇÏ¸é DB2´Â dbm°ú ´ÜÁö ¼ÒÀ½º ·¹°§¿¼-¸¸ ¼-·Î È£È¯¼ºÀÌ ÀÖ±â ¶§¹®¿¡, original dbm ÇüÅÂ(format)¸¦ Á÷Á¢ ´Ù·ê ¼ø±â ¶§¹®ÀÌ´Ù.

ndbmÀº Á»´õ »õ·Î¿î ÇüÅÂ·Î dbm¿¡ °ñÇØ ÈÎ¾À À¯¿¬ÇÏ´Ù. ±×·¯³ª ÀÌ°Í ¿ª½Ã dbmÀÌ °¡Áø ¿©·¯ °¡Áö ÇÑ°è(arbitrary limit)¸¦ ±×´ë·Î °¡Áö°í ÀÖ´Ù.

gdbmÀº GNU database managerÀÌ´Ù. ÀÌ°ÍÀº dbm°ú gdbm¿¡ ´ëÇÏ¿© È£È¯¼ºÀÌ ÀÖ´Ù.

db2´Â Sleepycat Software's DB2ÀÌ´Ù. ÀÌ°ÍÀº °¸Åë ´ÙÀ½°ú °°ÀÌ ¼³¸íµÈ´Ù. : "standalone°ú client/server ¼îÇÃ¸®ÄÉÀÌ¼ÇÀ» µ¿½Ã¿¡ Áö¿øÇÏ´Â °í¼º´ÉÀÇ ³»Åå µ¥ÀÌÅÍº£ÀÌ½º¸¦ Á¦°øÇØÁÖ´Â ÇÁ·Î±×·¥ toolkitÀÌ´Ù."("a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications.")

cdb´Â "³»¿ëÀÌ °¯ÇÏÁö ¾Ê´Â(constant) µ¥ÀÌÅÍº£¸¦ ¸¸µé°í ÀÐ¾î µéÀÌ´Âµ¥ °ü¸£°í, ¹ÏÀ½Á÷ÇÏ¸ç, °æ·®ÀÎ ÆÐÅ°ÁöºÀÌ´Ù. ÀÌ°ÍÀº qmailÀÇ Á¦ÀÛÀÚ¹¡ Á¦°øÇÏ°í ÀÖ°í, http://pobox.com/~djb/cdb.html¿¡¼- Ã£À» ¼ö ÀÖ´Ù. ÀÌ µ¥ÀÌÅÍº£ ´Â ³»¿ëÀÌ °¯ÇÏÁö ¾Ê´Â µ¥ÀÌÅÍ °£À½°ÀÌ¹Ç·Î, ÀÐ±â µ¿ÀÛ¸¸ Áö¿øÇÑ´Ù.

Example 1. DBA example

```
<?php
$id = dba_open("/tmp/test.db", "n", "db2");
if(!$id) {
    echo "dba_open failed\n";
    exit;
}
dba_replace("key", "This is an example!", $id);
if(dba_exists("key", $id)) {
    echo dba_fetch("key", $id);
    dba_delete("key", $id);
}
dba_close($id);
?>
```

DBA´Â binary safeÇÏ°í ¾î¶² ¾î¶² arbitrary limitµµ °¡Áö°í ÀÖÁö ¾Ê´Ù. ÀÌ°ÍÀÇ Á¦ÇÑÀº ±âÃÊ°¡ µÇ´Â µ¥ÀÌÅÍº£ÀÌ½º ±¸Çö¿¡ µû¸¥ ÇÑ°è¿¡¼- ¿Â °ÍÀÌ´Ù.

¸ðµç ÆÄÀÏ ±â¹Ý(file - based) µ¥ÀÌÅÍº£ÀÌ½º´Â »õ·Î ¸¸µé¾îÁø µ¥ÀÌÅÍº£ÀÌ½º¿¡ ´ëÇÏ¿© ÆÄÀÏ ¸ðµå¸¦ ÁöÁ¤ÇÏ´Â ¹æ¹ýÀ» Á¦°øÇØ ¾ß ÇÑ´Ù. ÀÌ¹ÝÀûÀ¸·Î ÆÄÀÏ ¸ðµå´Â **dba_open()**ÀÌ³ª **dba_popen()**ÀÇ ³× ¹øÂ° ÀÎ¼ö·Î Àü´ÞÇÏ°Ô µÈ´Ù.

¿©·¯ºÐÀº **dba_firstkey()**°ú **dba_nextkey()**ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ÀüÃ¼ µ¥ÀÌÅÍ¸¦ ¼øÂ÷ÀûÀ¸·Î Á¢±ÙÇÒ ¼ö ÀÖ´Ù. ±×¸®°í, µ¥ÀÌÅÍº£ÀÌ½º¸¦ Å½»öÇÏ´Â µ¿¾ÈÀº µ¥ÀÌÅÍº£ÀÌ½ºÀÇ ³×¿ëÀ» ¹Ù²Ü ¼ö ¾ø´Ù.

Example 2. Traversing a database

```
<?php
# ...open database...
$key = dba_firstkey($id);
while($key != false) {
    if(...) { # remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey($id);
}
?>
```

dba_close

dba_close - - µ¥ÀÌÅÍº£ÀÌ½º¸¦ ´Ý´Â´Ù.

Description

void dba_close(int handle);

dba_close() closes the established database and frees all resources specified by *handle*.

handle is a database handle returned by **dba_open()**.

dba_close() does not return any value.

See also: **dba_open()** **dba_popen()**

dba_delete

dba_delete - - ÁöÁ¤µÈ key°ªÀ» °¡Áö´Â entry¸¦ Áö¿î´Ù.

Description

string dba_delete(string key, int handle);

dba_delete() deletes the entry specified by *key* from the database specified with *handle*.

key is the key of the entry which is deleted.

handle is a database handle returned by **dba_open()**.

dba_delete() returns true or false, if the entry is deleted or not deleted, respectively.

See also: **dba_exists()** **dba_fetch()** **dba_insert()** **dba_replace()**

dba_exists

dba_exists - - ÁöÁ¤µÈ key°¡ ÀÖ´Â°¡ °Ë»çÇÑ´Ù.

Description

bool dba_exists(string key, int handle);

dba_exists() checks whether the specified *key* exists in the database specified by *handle*.

key is the key the check is performed for.

handle is a database handle returned by **dba_open()**.

dba_exists() returns true or false, if the key is found or not found, respectively.

See also: **dba_fetch()** **dba_delete()** **dba_insert()** **dba_replace()**

dba_fetch

dba_fetch - - ÁöÁ¤µÈ key°ªÀÇ data¸¦ °¡Á®¿Â´Ù.

Description

string dba_fetch(string key, int handle);

dba_fetch() fetches the data specified by *key* from the database specified with *handle*.

key is the key the data is specified by.

handle is a database handle returned by **dba_open()**.

dba_fetch() returns the associated string or false, if the key/data pair is found or not found, respectively.

See also: **dba_exists() dba_delete() dba_insert() dba_replace()**

dba_firstkey

dba_firstkey - - Ã¹ ¹øÂ° keyÀÇ °ªÀ» °¡Á®¿Â´Ù.

Description

string dba_firstkey(int handle);

dba_firstkey() returns the first key of the database specified by *handle* and resets the internal key pointer. This permits a linear search through the whole database.

handle is a database handle returned by **dba_open()**.

dba_firstkey() returns the key or false depending on whether it succeeds or fails, respectively.

See also: **dba_nextkey()**

dba_insert

dba_insert - - entry¸¦ »ðÀÔÇÑ´Ù.

Description

bool dba_insert(string key, string value, int handle);

dba_insert() inserts the entry described with *key* and *value* into the database specified by *handle*. It fails, if an entry with the same *key* already exists.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by **dba_open()**.

dba_insert() returns true or false, depending on whether it succeeds of fails, respectively.

See also: **dba_exists() dba_delete() dba_fetch() dba_replace()**

dba_nextkey

dba_nextkey - - ´ÙÀ½ key°ªÀ» °¡Á®¿Â´Ù.

Description

string dba_nextkey(int handle);

dba_nextkey() returns the next key of the database specified by *handle* and increments the internal key pointer.

handle is a database handle returned by **dba_open()**.

dba_nextkey() returns the key or false depending on whether it succeeds or fails, respectively.

See also: **dba_firstkey()**

dba_popen

dba_popen - - µ¥ÀÌÅÍº£ÀÌ½º¸¦ ¿µ±¸ÀûÀ¸·Î ¿¬´Ù.

Description

`int dba_popen`(string path, string mode, string handler, [...]);

dba_popen() establishes a persistent database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access, and "n" for truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_popen()** and can act on behalf of them.

dba_popen() returns a positive handler id or false, in the case the open is successful or fails, respectively.

See also: **dba_open() dba_close()**

dba_open

dba_open - - µ¥ÀÌÅÍº£ÀÌ½º¸¦ ¿¬´Ù.

Description

`int dba_open`(string path, string mode, string handler, [...]);

dba_open() establishes a database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access, and "n" for truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_open()** and can act on behalf of them.

dba_open() returns a positive handler id or false, in the case the open is successful or fails, respectively.

See also: **dba_popen() dba_close()**

dba_optimize

dba_optimize - - µ¥ÀÌÅÍº£ÀÌ½º¸¦ ÃÖÀûÈ-(Optimize)ÇÑ´Ù.

Description

`bool dba_optimize`(int handle);

dba_optimize() optimizes the underlying database specified by *handle*.

handle is a database handle returned by **dba_open()**.

dba_optimize() returns true or false, if the optimization succeeds or fails, respectively.

See also: **dba_sync()**

dba_replace

dba_replace - - entry¸¦ ´ëÅ¡(replace)ÇÏ°Å³ª »ðÀÔÇÑ´Ù.

Description

`bool dba_replace`(string key, string value, int handle);

dba_replace() replaces or inserts the entry described with *key* and *value* into the database specified by *handle*.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by **dba_open()**.

dba_replace() returns true or false, depending on whether it succeeds of fails, respectively.

See also: **dba_exists()** **dba_delete()** **dba_fetch()** **dba_insert()**

dba_sync

dba_sync - - µ¥ÀÌÅÍº£ÀÌ½º¸¦ µ¿±âÈ-(Synchronize)ÇÑ´Ù.

Description

bool dba_sync(int handle);

dba_sync() synchronizes the database specified by *handle*. This will probably trigger a physical write to disk, if supported.

handle is a database handle returned by **dba_open()**.

dba_sync() returns true or false, if the synchronization succeeds or fails, respectively.

See also: **dba_optimize()**

IX. dBase Functions

Table of Contents

dbase_create creates a dBase databas
dbase_open opens a dBase database
dbase_close close a dBase database
dbase_pack packs a dBase database
dbase_add_record add a record to a dBase database
dbase_delete_record deletes a record from a dBase database
dbase_get_record gets a record from a dBase database
dbase_numfields find out how many fields are in a dBase database
dbase_numrecords find out how many records are in a dBase database

ÀÌ ÇÔ¼öµéÀº dBase ÇüÅÂÀÇ databse(dbf)¿¡ ÀúÀåµÈ ·¹ÄÚµåµéÀ» Á¢±ÙÇÒ ¼ö ÀÖ°Ô ÇØ ÁØ´Ù.

index¿Í memo ÇÊµå´Â Áö¿øÇÏÁö ¾Ê´Â´Ù. ¶ÇÇÑ lockingµµ Áö¿øÇÏÁö ¾Ê´Â´Ù. µÎ °³ÀÇ À¥¼-¹ö ÇÁ·Î¼¼°¡ µ¿½Ã¿¡ °°Àº dBase ÆÄÀÏÀ» ¼öÁ¤ÇÏ·Á ÇÑ´Ù¸é, database ÀúÃ¼°¡ ¸Á°¡Áú ¼öµµ ÀÖ´Ù.

SQL µ¥ÀÌÅÍº£ÀÌ½º¿Í ´Ù¸£°Ô dBase µ¥ÀÌÅÍº£ÀÌ½º´Â »ý¼ºÈÄ¿¡ ±× ±¸Á¶¸¦ ¹Ù²Ü ¼ö ¾ø´Ù. ÇÑ ÆÄÀÏÀÌ »ý¼ºµÇ¸é ÇØ´ç µ¥ÀÌÅÍº£ÀÌ½ºÀÇ ¼±¾ðÀº °íÁ¤µÈ´Ù. ¼Óµµ Çâ»ó µîÀ» À§ÇØ »ç¿ëÇÏ´Â indexµµ Áö¿øÇÏÁö ¾Ê´Â´Ù. dBase´Â °íÁ¤µÈ ±æÀÌÀÇ ·¹ÄÚµå¸¦ °¡Áø ´Ü¼øÇÑ ¼øÂ÷ ÆÄÀÏÀÌ´Ù. »õ·¹ÄÚµå´Â ÆÄÀÏÀÇ ¸Ç µÚ¿¡ °Ù°í, »èÁ¦µÈ ·¹ÄÚµå´Â **dbase_pack()**ÀÌ ¼öÇàµÇ±â Àü¿¡´Â µ¥ÀÌÅÍ ÀÚÃ¼´Â À¯ÁöµÈ´Ù.

¿ì¸®´Â dBase ÆÄÀÏÀº »ç¿ëÇÏÁö ¸»±â¸¦ ±ÇÇÑ´Ù. ´ë½Å ÁøÂ¥ SQL ¼-¹ö¸¦ »ç¿ëÇÏ±â¸¦ ±ÇÇÑ´Ù. MySQLÀÌ³ª PostgreSQLÀÌ PHP¿Í °°ÀÌ ¸¹¹· »ç¿ëµÈ´Ù. dBase Áö¿øÀº ´ÜÁö ¿©·¯°ÐÀÌ »ç¿ëÇÏ´Â µ¥ÀÌÅÍº£ÀÌ½º¿¡ µ¥ÀÌÅÍ¸¦ ÀÐ¾î¼ÁÀ°Å³ª ³»¸¸³¾ ¶§¸¸ »ç¿ëÇÏ´Â °ÍÀÌ ÁÁ´Ù. dBase Æ÷¸ËÀº ´ë°Î°ÐÀÇ Windows¿ë ÇÁ·Î±×·¥ÀÌ »ç¿ë¸´ÉÇÑ µ¥ÀÌÅÍ Æ÷¸ËÀÌ±â ¶§¹®ÀÌ´Ù.

dbase_create

dbase_create - - dBase database¸¦ »ý¼ºÇÑ´Ù.

Description

int dbase_create(string filename, array fields);

The *fields* parameter is an array of arrays, each array describing the format of one field in the database. Each field consists of a name, a character indicating the field type, a length, and a precision.

The types of fields available are:

L - Boolean. These do not have a length or precision.

M - Memo. (Note that these aren't supported by PHP.) These do not have a length or precision.

D - Date (stored as YYYYMMDD). These do not have a length or precision.

N - Number. These have both a length and a precision (the number of digits after the decimal point).

C - String.

If the database is successfully created, a dbase_identifier is returned, otherwise false is returned.
Example 1. Creating a dBase database file

```
// "database" name
$dbname = "/tmp/test.dbf";
// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );
// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_open

dbase_open - - dBase database¸¦ ¿¬´Ù.

Description

int dbase_open(string filename, int flags);

The flags correspond to those for the open() system call. (Typically 0 means read - only, 1 means write - only, and 2 means read and write.)

Returns a dbase_identifier for the opened database, or false if the database couldn't be opened.

dbase_close

dbase_close - - dBase database¸¦ ´Ý´Â´Ù.

Description

bool dbase_close(int dbase_identifier);

Closes the database associated with *dbase_identifier*.

dbase_pack

dbase_pack - - dBase database¸¦ packÇÑ´Ù.

Description

bool dbase_pack(int dbase_identifier);

Packs the specified database (permanently deleting all records marked for deletion using **dbase_delete_record()**.

dbase_add_record

dbase_add_record - - dBase database¿¡ ÇÑ record¸¦ ´õÇÑ´Ù.

Description

bool dbase_add_record(int dbase_identifier, array record);

Adds the data in the *record* to the database. If the number of items in the supplied record isn't equal to the number of fields in the database, the operation will fail and false will be returned.

dbase_delete_record

dbase_delete_record - - dBase database¿¡¼- ÇÑ record¸¦ »èÁ¦ÇÑ´Ù

Description

bool dbase_delete_record(int dbase_identifier, int record);

Marks *record* to be deleted from the database. To actually remove the record from the database, you must also call **dbase_pack()**.

dbase_get_record

dbase_get_record - - dBase database¿¡¼- ÇÑ record¸¦ ÀÐ¾î¿Â´Ù.

Description

array dbase_get_record(int dbase_identifier, int record);

Returns the data from *record* in an array. The array is indexed starting at 0, and includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see **dbase_delete_record()**.

Each field is converted to the appropriate PHP type. (Dates are left as strings.)

dbase_numfields

dbase_numfields - - dBase databaseÀÇ fieldÀÇ °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int dbase_numfields(int dbase_identifier);

Returns the number of fields (columns) in the specified database. Field numbers are between 0 and dbase_numfields($db) - 1, while record numbers are between 1 and dbase_numrecords($db).
Example 1. Using dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords

dbase_numrecords - - fdBase databaseÀÇ recordÀÇ °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int dbase_numrecords(int dbase_identifier);

Returns the number of records (rows) in the specified database. Record numbers are between 1 and dbase_numrecords($db), while field numbers are between 0 and dbase_numfields($db) - 1.

X. dbm Functions

Table of Contents

dbmnextkey
dblist

ÀÌ ÇÔ¼öµéÀº dbm ÇüÅÂÀÇ databse(dbf)¿¡ ÀúÀåµÈ ·¹ÄÚµåµéÀ» Á¢±ÙÇÒ ¼ö ÀÖ°Ô ÇØ ÁØ´Ù. ÀÌ ÇüÅÂÀÇ database´Â Berkeley db, gdbm µî°ú ³»Åå¸È flatfile ¶óÀÌºê·¯¸® °°À° ÀÏ°Î system ¶óÀÌºê·¯¸® µîÀÌ Áö¿øÇÏ´Âµ¥, ÀÏ¹ÝÀûÀÎ relational databases¿Í ´Þ¸® key/valueÀÇ ½ÖÀ¸·Î data¸¦ ÀúÀåÇÑ´Ù.
Example 1. dbm example

```
$dbm = dbmopen("lastseen", "w");
if (dbmexists($dbm, $userid)) {
  $last_seen = dbmfetch($dbm, $userid);
} else {
  dbminsert($dbm, $userid, time());
}
do_stuff();
dbmreplace($dbm, $userid, time());
dbmclose($dbm);
```

dbmopen

dbmopen - - dbm database¸¦ ¿¬´Ù.

Description

int dbmopen(string filename, int flags);

The first argument is the full-path filename of the dbm file to be opened and the second is the file open mode which is one of "r", "n" or "w" for read, new (implies write) and write respectively.

Returns an identifer to be passed to the other dbm functions on success, or false on failure.

If ndbm support is used, ndbm will actually create filename.dir and filename.pag files. gdbm only uses one file, as does the internal flat-file support, and Berkeley db creates a filename.db file. Note that PHP does its own file locking in addition to any file locking that may be done by the dbm library itself. PHP does not delete the .lck files it creates. It uses these files simply as fixed inodes on which to do the file locking. For more information on dbm files, see your Unix man pages, or obtain GNU's gdbm from ftp://prep.ai.mit.edu/pub/gnu.
Example 1. dbm example

```
$dbm = dbmopen("lastseen", "w");
if (dbmexists($dbm, $userid)) {
  $last_seen = dbmfetch($dbm, $userid);
} else {
  dbminsert($dbm, $userid, time());
}
do_stuff();
dbmreplace($dbm, $userid, time());
dbmclose($dbm);
```

dbmclose

dbmclose - - dbm database¸¦ ´Ý´Â´Ù.

Description

bool dbmclose(int dbm_identifier);

Unlocks and closes the specified database.

dbmexists

dbmexists - - dbm database¿¡ ÁÖ¾îÁø key¿¡ ÇØ´çÇÏ´Â °ªÀÌ ÀÖ´ÂÁö ¾Ë¾Æ¿»´Ù.

Description

bool dbmexists(int dbm_identifier, string key);

Returns true if there is a value associated with the *key*.

dbmfetch

dbmfetch - - dbm database¿¡¼- ÁÖ¾îÁø keyÀÇ value¸¦ ÀÐ¾î¿Â´Ù.

Description

string dbmfetch(int dbm_identifier, string key);

Returns the value associated with *key*.

dbminsert

dbminsert - - dbm database¿¡ keyÀÇ value¸¦ »ðÀÔÇÑ´Ù.

Description

int dbminsert(int dbm_identifier, string key, string value);

Adds the value to the database with the specified key.

Returns -1 if the database was opened read-only, 0 if the insert was successful, and 1 if the specified key already exists. (To replace the value, use **dbmreplace()**.)

dbmreplace

dbmreplace - - dbm database¿¡¼- keyÀÇ value¸¦ ¹Ù²Û´Ù.

Description

bool dbmreplace(int dbm_identifier, string key, string value);

Replaces the value for the specified key in the database.

This will also add the key to the database if it didn't already exist.

dbmdelete

dbmdelete - - dbm database¿¡¼- keyÀÇ value¸¦ »èÁ¦ÇÑ´Ù.

Description

bool dbmdelete(int dbm_identifier, string key);

Deletes the value for *key* in the database.

Returns false if the key didn't exist in the database.

dbmfirstkey

dbmfirstkey - - dbm databaseÀÇ Ã¹ ¹øÂ° key¸¦ °Ë»öÇÑ´Ù.

Description

string dbmfirstkey(int dbm_identifier);

Returns the first key in the database. Note that no particular order is guaranteed since the database may be built using a hash-table, which doesn't guarantee any ordering.

dbmnextkey

dbmnextkey - - dbm databaseÀÇ ´ÙÀ½ key¸¦ °Ë»öÇÑ´Ù.

Description

string dbmnextkey(int dbm_identifier, string key);

Returns the next key after *key*. By calling **dbmfirstkey()** followed by successive calls to **dbmnextkey()** it is possible to visit every key/value pair in the dbm database. For example:
Example 1. Visiting every key/value pair in a dbm database.

```
$key = dbmfirstkey($dbm_id);
```

```
while ($key) {
    echo "$key = " . dbmfetch($dbm_id, $key) . "\n";
    $key = dbmnextkey($dbm_id, $key);
}
```

dblist

dblist - - »ç¿ëÁßÀÎ dbm - compatible library¿¡ ´ëÇÑ Á¤º¸¸¦ Ç¥½ÃÇÑ´Ù.

Description

string dblist(void);

XI. Directory Functions

Table of Contents

chdir

chdir - - ÇöÀç directory¸¦ ¹Ù²Û´Ù.

Description

int chdir(string directory);

Changes PHP's current directory to *directory*. Returns FALSE if unable to change directory, TRUE otherwise.

dir

dir - - directory class

Description

new dir(string directory);

A pseudo - object oriented mechanism for reading a directory. The given *directory* is opened. Two properties are available once directory has been opened. The handle property can be used with other directory functions such as **readdir()**, **rewinddir()** and **closedir()**. The path property is set to path the directory that was opened. Three methods are available: read, rewind and close.

Example 1. Dir() Example

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while($entry=$d->read()) {
    echo $entry."<br>\n";
}
$d->close();
```

closedir

closedir - - directory handleÀ» ´Ý´Â´Ù.

Description

void closedir(int dir_handle);

Closes the directory stream indicated by *dir_handle*. The stream must have previously been opened by **opendir()**.

opendir

opendir - - directory handleÀ» ¿¬´Ù.

Description

int opendir(string path);

Returns a directory handle to be used in subsequent **closedir()**, **readdir()**, and **rewinddir()** calls.

readdir

readdir - - directory handle·Î °ÎÅÍ Ç×¸ñÀ» ÀÐ¾î¿Â´Ù.

Description

string readdir(int dir_handle);

Returns the filename of the next file from the directory. The filenames are not returned in any particular order.
Example 1. List all files in the current directory

```
<?php
    $handle=opendir('.');
    echo "Directory handle: $handle\n";
    echo "Files:\n";
    while ($file = readdir($handle)) {
        echo "$file\n";
    }
    closedir($handle);
?>
```

rewinddir

rewinddir - - directory handleÀ» directoryÀÇ ½ÃÀÛ À§Ä¡·Î µÇµ¹¸°´Ù.

Description

void rewinddir(int dir_handle);

Resets the directory stream indicated by *dir_handle* to the beginning of the directory.

XII. Dynamic Loading Functions

Table of Contents

dl

dl - - PHP extensionÀ» ½ÇÇàÁß¿¡ loadÇÑ´Ù.

Description

int dl(string library);

Loads the PHP extension defined in *library*. See also the extension_dir configuration directive.

XIII. Program Execution Functions

Table of Contents

escapeshellcmd

escapeshellcmd - - shellÀÇ metacharacterµéÀ» escapeÇÑ´Ù.

Description

string escapeshellcmd(string command);

EscapeShellCmd() escapes any characters in a string that might be used to trick a shell command into executing arbitrary commands. This function should be used to make sure that any data coming from user input is escaped before this data is passed to the **exec()** or **system()** functions. A standard use would be:

```
system(EscapeShellCmd($cmd))
```

exec

exec - - ¿Ü°Î programÀ» ½ÇÇàÇÑ´Ù.

Description

string exec(string command, string array, int return_var);

Exec executes the given *command*, however it does not output anything. It simply returns the last line from the result of the command. If you need to execute a command and have all the data from the command passed directly back without any interference, use the **PassThru()** function.

If the *array* argument is present, then the specified array will be filled with every line of output from the command. Note that if the array already contains some elements, **exec()** will append to the end of the array. If you do not want the function to append elements, call **unset()** on the array before passing it to **exec()**.

If the *return_var* argument is present along with the *array* argument, then the return status of the executed command will be written to this variable.

Note that if you are going to allow data coming from user input to be passed to this function, then you should be using **EscapeShellCmd()** to make sure that users cannot trick the system into executing arbitrary commands.

See also **system()**, **PassThru()**, **popen()** and **EscapeShellCmd()**.

system

system - - ¿Ü°Î programÀ» ½ÇÇàÇÏ°í ±× °á°ú¸¦ Ãâ·ÂÇÑ´Ù.

Description

string system(string command, int return_var);

System() is just like the C version of the function in that it executes the given *command* and outputs the result. If a variable is provided as the second argument, then the return status code of the executed command will be written to this variable.

Note, that if you are going to allow data coming from user input to be passed to this function, then you should be using the **EscapeShellCmd()** function to make sure that users cannot trick the system into executing arbitrary commands.

The **System()** call also tries to automatically flush the web server's output buffer after each line of output if PHP is running as a server module.

If you need to execute a command and have all the data from the command passed directly back without any interference, use the **PassThru()** function. See also the **exec()** and **popen()** functions.

passthru

passthru - - ¿Ü°Î ÇÁ·Î±×·¥À» ½ÇÇàÇÏ°í ±× °á°ú¸¦ Á÷Á¢ (ÁöÃ¼¸øÀÌ) Ãâ·ÂÇÑ´Ù.

Description

string passthru(string command, int return_var);

The PassThru() function is similar to the **Exec()** function in that it executes a *command*. If the *return_var* argument is present, the return status of the Unix command will be placed here. This function should be used in place of **Exec()** or **System()** when the output from the Unix command is binary data which needs to be passed directly back to the browser. A common use for this is to execute something like the pbmplus utilities that can output an image stream directly. By setting the content-type to *image/gif* and then calling a pbmplus program to output a gif, you can create PHP scripts that output images directly.

See also **exec()** and **fpassthru()**.

XIV. filePro Functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº filePro ÇüÅÂÀÇ databse(dbf)¿¡ ÀúÀåµÈ ·¹ÄÚµåµéÀ» read - only·Î Á¢±ÙÇÒ ¼ö ÀÖ°Ô ÇØ ÁØ´Ù.

filePro´Â Fiserv, Inc.ÀÇ trademark·Î µî·ÏµÇ¾î ÀÖ´Ù. ¿©·¯°ÐÀº http://www.fileproplus.com/¿¡¼ filePro¿¡ °üÇÑ ´õ ¸¹Àº Á¤º¸¸¦ ¾òÀ» ¼ö ÀÖ´Ù.

filepro

filepro - - map fileÆÄÀÏÀ» ÀÐ°í °ËÁõÇÑ´Ù.

Description

bool filepro(string directory);

This reads and verifies the map file, storing the field count and info.

No locking is done, so you should avoid modifying your filePro database while it may be opened in PHP.

filepro_fieldname

filepro_fieldname - - fieldÀÇ ÀÌ¸§À» ±¸ÇÑ´Ù.

Description

string filepro_fieldname(int field_number);

Returns the name of the field corresponding to *field_number*.

filepro_fieldtype

filepro_fieldtype - - fieldÀÇ typeÀ» ±¸ÇÑ´Ù.

Description

string filepro_fieldtype(int field_number);

Returns the edit type of the field corresponding to *field_number*.

filepro_fieldwidth

filepro_fieldwidth - - fieldÀÇ Å©±â(width)¸¦ ±¸ÇÑ´Ù.

Description

int filepro_fieldwidth(int field_number);

Returns the width of the field corresponding to *field_number*.

filepro_retrieve

filepro_retrieve - - filePro database¿¡¼ µ¥ÀÌÅ¸¸¦ °Ë»öÇÑ´Ù.

Description

string filepro_retrieve(int row_number, int field_number);

Returns the data from the specified location in the database.

filepro_fieldcount

filepro_fieldcount - - filePro databaseÀÇ fieldÀÇ °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int filepro_fieldcount(void);

Returns the number of fields (columns) in the opened filePro database.

See also **filepro()**.

filepro_rowcount

filepro_rowcount - - filePro databaseÀÇ rowÀÇ °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int filepro_rowcount(void);

Returns the number of rows in the opened filePro database.

See also **filepro()**.

XV. Filesystem Functions

Table of Contents

basename

basename - - pathÁß name ºÎºÐÀ» ±¸ÇÑ´Ù.

Description

string basename(string path);

Given a string containing a path to a file, this function will return the base name of the file.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).
Example 1. basename() example

```
$path = "/home/httpd/html/index.php3";
$file = basename($path); // $file is set to "index.php3"
```

See also: **dirname()**

chgrp

chgrp - - ÆÄÀÏÀÇ groupÀ» ¹Ù²Û´Ù.

Description

int chgrp(string filename, mixed group);

Attempts to change the group of the file filename to group. Only the superuser may change the group of a file arbitrarily; other users may change the group of a file to any group of which that user is a member.

Returns true on success; otherwise returns false.

On Windows, does nothing and returns true.

See also **chown()** and **chmod()**.

chmod

chmod - - ÆÄÀÏÀÇ mode¸¦ ¹Ù²Û´Ù.

Description

int chmod(string filename, int mode);

Attempts to change the mode of the file specified by *filename* to that given in *mode*.

Note that *mode* is not automatically assumed to be an octal value. To ensure the expected operation, you need to prefix *mode* with a zero (0):

```
chmod( "/somedir/somefile", 755 );   // decimal; probably incorrect
chmod( "/somedir/somefile", 0755 );  // octal; correct value of mode
```

Returns true on success and false otherwise.

See also **chown()** and **chgrp()**.

chown

chown - - ÆÄÀÏÀÇ owner¸¦ ¹Ù²Û´Ù.

Description

int chown(string filename, mixed user);

Attempts to change the owner of the file filename to user user. Only the superuser may change the owner of a file.

Returns true on success; otherwise returns false.

> **NOTE:** On Windows, does nothing and returns true.

See also **chown()** and **chmod()**.

clearstatcache

clearstatcache - - ÆÄÀÏÀÇ stat cache¸¦ ¹Ù²Û´Ù.

Description

void clearstatcache(void);

Invoking the stat() or lstat() system call on most systems is quite expensive. Therefore, the result of the last call to any of the status functions (listed below) is stored for use on the next such call using the same filename. If you wish to force a new status check, for instance if the file is being checked many times and may change or disappear, use this function to clear the results of the last call from memory.

This value is only cached for the lifetime of a single request.

Affected functions include **stat()**, **lstat()**, **file_exists()**, **is_writeable()**, **is_readable()**, **is_executable()**, **is_file()**, **is_dir()**, **is_link()**, **filectime()**, **fileatime()**, **filemtime()**, **fileinode()**, **filegroup()**, **fileowner()**, **filesize()**, **filetype()**, and **fileperms()**.

copy

copy - - ÆÄÀÏÀ» º¹»çÇÑ´Ù.

Description

int copy(string source, string dest);

Makes a copy of a file. Returns true if the copy succeeded, false otherwise.
Example 1. copy() example

```
if (!copy($file, $file.'.bak')) {
    print("failed to copy $file...<br>\n");
}
```

See also: **rename()**

delete

delete - - ½ÇÁ¦·Î´Â ¾ø´Â ¸í·É

Description

void delete(string file);

ÀÌ ¸í·ÉÀ» Ã£´Â »ç¶÷µéÀº ¾Æ¸¶ **unlink()**³ª **unset()** ¸í·ÉÀ» »ç¿ëÇÏ´Â ±â´ÉÀ» ¿øÇÒ °ÍÀÌ´Ù.

See also: ÆÄÀÏÀ» Áö¿ì±â À§ÇØ¼ Â **unlink()**, °¡¼ö¸¦ Áö¿ì±â À§ÇØ¼ Â **unset()**

dirname

dirname - - pathÀÇ Directory°Î°ÐÀ» µ¹·ÁØ´Ù.

Description

string dirname(string path);

Given a string containing a path to a file, this function will return the name of the directory.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).
Example 1. dirname() example

```
$path = "/etc/passwd";
$file = dirname($path); // $file is set to "/etc"
```

See also: **basename()**

diskfreespace

diskfreespace - - ÇØ´ç µð·°Ää¸®¿¡ ÀúÀå °¡´ÉÇÑ ³²Àº ¿ë·®À» ¹ÝÈ¯ÇÑ´Ù.

Description

`float diskfreespace`(string directory);

Given a string containing a directory, this function will return the number of bytes available on the corresponding disk.
Example 1. diskfreespace() example

```
$df = diskfreespace("/"); // $df contains the number of bytes available on "/"
```

fclose

fclose - - ÁöÁ¤µÈ file pointer¸¦ ´Ý´Â´Ù.

Description

int fclose(int fp);

The file pointed to by fp is closed.

Returns true on success and false on failure.

The file pointer must be valid, and must point to a file successfully opened by **fopen()** or **fsockopen()**.

feof

feof - - file pointer°¡ end-of-file¿¡ ÀÖ´Â°¡ °Ë»çÇÑ´Ù.

Description

int feof(int fp);

Returns true if the file pointer is at EOF or an error occurs; otherwise returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

fgetc

fgetc - - file pointer¿¡¼ ¹®ÀÚ¸¦ ÀÐ´Â´Ù.

Description

string fgetc(int fp);

Returns a string containing a single character read from the file pointed to by fp. Returns FALSE on EOF (as does **feof()**).

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

See also **fopen()**, **popen()**, **fsockopen()**, and **fgets()**.

fgetcsv

fgetcsv - - ÆÄÀÏÀÇ ÇöÀç À§Ä¡¿¡¼- CSV ÇÊµå·Î µÈ ÁÙÀ» ÀÐ¾î¿Í¼- parseÇÑ´Ù.

Description

array fgetcsv(int fp, int length);

Similar to fgets() except that fgetcsv() parses the line it reads for fields in CSV format and returns an array containing the fields read.

fp must be a valid file pointer to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**

length must be greater than the longest line to be found in the CSV file (allowing for trailing line - end characters).

fgetcsv() returns false on error, including end of file.

NB A blank line in a CSV file will be returned as an array comprising just one single null field, and will not be treated as an error.
Example 1. fgetcsv() example - Read and print entire contents of a CSV file

```
$row=1;
$fp = fopen("test.csv","r");
while ($data = fgetcsv($fp,1000)) {
        $num = count($data);
        print "<p> $num fields in line $row: <br>";
        $row++;
        for ( $c=0; $c<$num; $c++ ) print $data[$c] . "<br>";
}
fclose($fp);
```

fgets

fgets - - file pointer¿¡¼- ÇÑ ÁÙÀ» ÀÐ¾î ¿Â´Ù.

Description

string fgets(int fp, int length);

Returns a string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

See also **fopen()**, **popen()**, **fgetc()**, and **fsockopen()**.

fgetss

fgetss - - file pointer¿¡¼- ÇÑ ÁÙÀ» ÀÐ¾î¿Í HTML tagµéÀ» stripÇÑ´Ù.

Description

string fgetss(int fp, int length);

Identical to **fgets()**, except that fgetss attempts to strip any HTML and PHP tags from the text it reads.

See also **fgets()**, **fopen()**, **fsockopen()**, and **popen()**.

file

file - - ÆÄÀÏ ÀüÃ¼¸¦ ÀÐ¾î¼- ¹è¿-¿¡ ÀúÀåÇÑ´Ù.

Description

array file(string filename);

Identical to **readfile()**, except that file() returns the file in an array.

See also **readfile()**, **fopen()**, and **popen()**.

file_exists

file_exists - - ÆÄÀÏÀÌ Á¸ÀçÇÏ´ÂÁö °Ë»çÇÑ´Ù.

Description

int file_exists(string filename);

Returns true if the file specified by *filename* exists; false otherwise.

See also **clearstatcache()**.

fileatime

fileatime - - ÆÄÀÏ¿¡ ¸¶Áö¸·À¸·Î Á¢±ÙÇÑ ½Ã°£À» ±¸Ç´Ù.

Description

int fileatime(string filename);

Returns the time the file was last accessed, or false in case of an error.

filectime

filectime - - ÆÄÀÏÀÇ inode°¡ º¯°æµÈ ½Ã°£À» ±¸Ç´Ù.

Description

int filectime(string filename);

Returns the time the file was last changed, or false in case of an error.

filegroup

filegroup - - ÆÄÀÏÀÇ groupÀ» ±¸Ç´Ù.

Description

int filegroup(string filename);

Returns the group ID of the owner of the file, or false in case of an error.

fileinode

fileinode - - ÆÄÀÏÀÇ inode¸¦ ±¸Ç´Ù.

Description

int fileinode(string filename);

Returns the inode number of the file, or false in case of an error.

filemtime

filemtime - - ÆÄÀÏÀÌ ¼öÁ¤µÈ ½Ã°£À» ±¸Ç´Ù.

Description

int filemtime(string filename);

Returns the time the file was last modified, or false in case of an error.

fileowner

fileowner - - ÆÄÀÏÀÇ owner¸¦ ±¸Ñ´Ù.

Description

int fileowner(string filename);

Returns the user ID of the owner of the file, or false in case of an error.

fileperms

fileperms - - ÆÄÀÏÀÇ permissionÀ» ±¸Ñ´Ù.

Description

int fileperms(string filename);

Returns the permissions on the file, or false in case of an error.

filesize

filesize - - ÆÄÀÏÀÇ Å©±â¸¦ ±¸Ñ´Ù.

Description

int filesize(string filename);

Returns the size of the file, or false in case of an error.

filetype

filetype - - ÆÄÀÏÀÇ typeÀ» ±¸Ñ´Ù.

Description

string filetype(string filename);

Returns the type of the file. Possible values are fifo, char, dir, block, link, file, and unknown.

Returns false if an error occurs.

flock

flock - - portable ÆÄÀÏ locking

Description

`bool flock`(int fp, int operation);

PHP supports a portable way of locking whole files in an advisory way (which means all accessing programs have to use the same way of locking or it will not work).

To lock a file, you have to open it first and pass the resulting file pointer to flock. The operations you can use are 1 for locking the file as a reader (shared lock), 2 for locking the file as a writer (exclusive lock) and 3 for releasing a lock. This enables a simple reader/writer model which can be used on almost every platform (including most Unixes and even Windows).

If you add 4 to the operation parameter, locking will be tried in a non - blocking way. It returns false, if the lock cannot be acquired non - blocking. Otherwise, flock will always return true.

fopen

fopen - - ÆÄÀÏÀÌ³ª URLÀ» ¿¬´Ù.

Description

int fopen(string filename, string mode);

If *filename* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server and a file pointer is returned to the beginning of the text of the response.

Does not handle HTTP redirects, so you must include trailing slashes on directories.

If *filename* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and a pointer to the requested file is returned. If the server does not support passive mode ftp, this will fail. You can open files for either reading and writing via ftp (but not both simultaneously).

If *filename* begins with anything else, the file will be opened from the filesystem, and a file pointer to the file opened is returned.

If the open fails, the function returns false.

mode may be any of the following:

'r' - Open for reading only; place the file pointer at the beginning of the file.
'r+' - Open for reading and writing; place the file pointer at the beginning of the file.
'w' - Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'w+' - Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'a' - Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
'a+' - Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

As well, *mode* may contain the letter 'b'. This is useful only on systems which differentiate between binary and text files (i.e., it's useless on Unix). If not needed, this will be ignored.
Example 1. fopen() example

```
$fp = fopen("/home/rasmus/file.txt", "r");
$fp = fopen("http://www.php.net/", "r");
$fp = fopen("ftp://user:password@example.com/", "w");
```

If you are experiencing problems with reading and writing to files and you're using the server module version of PHP, remember to make sure that the files and directories you're using are accessible to the server process.

On the Windows platform, be careful to escape any backslashes used in the path to the file, or use forward slashes.

```
$fp = fopen("c:\\data\\info.txt", "r");
```

See also **fclose()**, **fsockopen()**, and **popen()**.

fpassthru

fpassthru - - file pointer¿¡¼-°ÎÅÍ ³²¾Æ ÀÖ´Â ¸ðµç µ¥ÀÌÅ¸¸¦ Ãâ·ÂÇÑ´Ù.

Description

int fpassthru(int fp);

Reads to EOF on the given file pointer and writes the results to standard output.

If an error occurs, **fpassthru()** returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**. The file is closed when **fpassthru()** is done reading it (leaving *fp* useless).

If you just want to dump the contents of a file to stdout you may want to use the **readfile()**, which saves you the **fopen()** call.

See also **readfile()**, **fopen()**, **popen()**, and **fsockopen()**

fputs

fputs - - file pointer¿¡ ¾´´Ù.

Description

int fputs(int fp, string str, int [length]);

fputs() is an alias to **fwrite()**, and is identical in every way. Note that the *length* parameter is optional and if not specified the entire string will be written.

fread

fread - - ÆÄÀÏÀ» binary·Î ÀÐ¾î¿Â´Ù.

Description

string fread(int fp, int length);

fread() reads up to *length* bytes from the file pointer referenced by *fp*. Reading stops when *length* bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a file into a string
$filename = "/usr/local/something.txt";
$fd = fopen( $filename, "r" );
$contents = fread( $fd, filesize( $filename ) );
fclose( $fd );
```

See also **fwrite()**, **fopen()**, **fsockopen()**, **popen()**, **fgets()**, **fgetss()**, **file()**, and **fpassthru()**.

fseek

fseek - - file pointer¸¦ ¿Å±ä´Ù.

Description

int fseek(int fp, int offset);

Sets the file position indicator for the file referenced by fp to offset bytes into the file stream. Equivalent to calling (in C) `fseek( fp, offset, SEEK_SET )`.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

May not be used on file pointers returned by **fopen()** if they use the "http://" or "ftp://" formats.

See also **ftell()** and **rewind()**.

ftell

ftell - - file pointerÀÇ read/write À§Ä¡¸¦ ±¸Ñ´Ù.

Description

int ftell(int fp);

Returns the position of the file pointer referenced by fp; i.e., its offset into the file stream.

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **fopen()** or **popen()**.

See also **fopen()**, **popen()**, **fseek()** and **rewind()**.

fwrite

fwrite - - ÆÄÀÏÀ» Binary·Î ¾´´Ù.

Description

int fwrite(int fp, string string, int length);

fwrite() writes the contents of *string* to the file stream pointed to by *fp*. If the *length* argument is given, writing will stop after *length* bytes have been written or the end of *string* is reached, whichever comes first.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

See also **fread()**, **fopen()**, **fsockopen()**, **popen()**, and **fputs()**.

set_file_buffer

set_file_buffer - - ÇØ´ç ÆÄÀÏ Æ÷ÀÎÅÍ¿¡¼- ÆÄÀÏ ¹öÆÛ¸µ(buffering)À» ÁöÁ¤ÇÑ´Ù.

Description

int fwrite(int fp, int buffer);

set_file_buffer() sets the buffering for write operations on the given filepointer *fp* to *buffer* bytes. If *buffer* is 0 then write operations are unbuffered.

The function returns 0 on success, or EOF if the request cannot be honored.

Note that the default for any fopen with calling set_file_buffer is 8K.

See also **fopen()**.

is_dir

is_dir - - ÁöÁ¤µÈ ÆÄÀÏ¸íÀÌ directoryÀÎ°¡ ¾Ë·ÁÁØ´Ù.

Description

bool is_dir(string filename);

Returns true if the filename exists and is a directory.

See also **is_file()** and **is_link()**.

is_executable

is_executable - - ÁöÁ¤µÈ ÆÄÀÏ¸íÀÌ ½ÇÇà°¡´É ÆÄÀÏÀÎ°¡ ¾Ë·ÁÁØ´Ù.

Description

bool is_executable(string filename);

Returns true if the filename exists and is executable.

See also **is_file()** and **is_link()**.

is_file

is_file - - ÁöÁ¤µÈ ÆÄÀÏ¸íÀÌ °¡ÀëÆÄÀÏÀÎ°¡ ¾Ë·ÁÁØ´Ù.

Description

bool is_file(string filename);

Returns true if the filename exists and is a regular file.

See also **is_dir()** and **is_link()**.

is_link

is_link - - ÁöÁ¤µÈ ÆÄÀÏ¸íÀÌ symbolic linkÀÎ°¡ ¾Ë·ÁÁØ´Ù.

Description

bool is_link(string filename);

Returns true if the filename exists and is a symbolic link.

See also **is_dir()** and **is_file()**.

is_readable

is_readable - - ÁöÁ¤µÈ ÆÄÀÏ¸íÀÌ ÀÐ±â °¡´ÉÀÎ°¡ ¾Ë·ÁÁØ´Ù.

Description

bool is_readable(string filename);

Returns true if the filename exists and is readable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

See also **is_writeable()**.

is_writeable

is_writeable - - ÁöÁ¤µÈ ÆÄÀÏ¸íÀÌ ¾²±â °¡´ÉÀÎ°¡ ¾Ë·ÁÁØ´Ù.

Description

bool is_readable(string filename);

Returns true if the filename exists and is writeable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

See also **is_readable()**.

link

link - - hard link¸¦ ¸¸µç´Ù.

Description

int link(string target, string link);

Link() creates a hard link.

See also the **symlink()** to create soft links, and **readlink()** along with **linkinfo()**.

linkinfo

linkinfo - - link¿¡ ´ëÇÑ Á¤º¸¸¦ ±¸Ç´Ù.

Description

int linkinfo(string path);

Linkinfo() returns the st_dev field of the UNIX C stat structure returned by the lstat system call. This function is used to verify if a link (pointed to by *path*) really exists (using the same method as the S_ISLNK macro defined in stat.h). Returns 0 or FALSE in case of error.

See also **symlink()**, **link()**, and **readlink()**.

mkdir

mkdir - - directory¸¦ ¸¸µç´Ù.

Description

int mkdir(string pathname, int mode);

Attempts to create the directory specified by pathname.

Note that you probably want to specify the mode as an octal number, which means it should have a leading zero.

```
mkdir("/path/to/my/dir", 0700);
```

Returns true on success and false on failure.

See also **rmdir()**.

pclose

pclose - - process file pointer¸¦ ´Ý´Â´Ù.

Description

int pclose(int fp);

Closes a file pointer to a pipe opened by **popen()**.

The file pointer must be valid, and must have been returned by a successful call to **popen()**.

Returns the termination status of the process that was run.

See also **popen()**.

popen

popen - - process file pointer¸¦ ¿¬´Ù.

Description

int popen(string command, string mode);

Opens a pipe to a process executed by forking the command given by command.

Returns a file pointer identical to that returned by **fopen()**, except that it is unidirectional (may only be used for reading or writing) and must be closed with **pclose()**. This pointer may be used with **fgets()**, **fgetss()**, and **fputs()**.

If an error occurs, returns false.

```
$fp = popen( "/bin/ls", "r" );
```

See also **pclose()**.

readfile

readfile - - fileÀ» ÀÐ¾î Ãâ·ÂÇÑ´Ù.

Description

int readfile(string filename);

Reads a file and writes it to standard output.

Returns the number of bytes read from the file. If an error occurs, false is returned and unless the function was called as @readfile, an error message is printed.

If *filename* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server and the text of the response is written to standard output.

Does not handle HTTP redirects, so you must include trailing slashes on directories.

If *filename* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and the requested file is written to standard output. If the server does not support passive mode ftp, this will fail.

If *filename* begins with neither of these strings, the file will be opened from the filesystem and its contents written to standard output.

See also **fpassthru()**, **file()**, **fopen()**, **include()**, **require()**, and **virtual()**.

readlink

readlink - - symbolic linkÀÇ ´ë»óÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

int readlink(string path);

Readlink() does the same as the readlink C function and returns the contents of the symbolic link path or 0 in case of error.

See also **symlink()**, **readlink()** and **linkinfo()**.

rename

rename - - ÆÄÀÏ¸íÀ» ¹Ù²Û´Ù.

Description

int rename(string oldname, string newname);

Attempts to rename *oldname* to *newname*.

Returns true on success and false on failure.

rewind

rewind - - file pointer¸¦ ÆÄÀÏÀÇ Ã³À½ À§Ä¡·Î ¼³Á¤ÇÑ´Ù.

Description

int rewind(int fp);

Sets the file position indicator for fp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**.

See also **fseek()** and **ftell()**.

rmdir

rmdir - - directory¸¦ Áö¿î´Ù.

Description

int rmdir(string dirname);

Attempts to remove the directory named by pathname. The directory must be empty, and the relevant permissions must permit this.

If an error occurs, returns 0.

See also **mkdir()**.

stat

stat - - ÆÄÀÏÀÇ stat Á¤º¸¸¦ ±¸ÇÑ´Ù.

Description

array stat(string filename);

Gathers the statistics of the file named by filename.

Returns an array with the statistics of the file with the following elements:

device
inode
number of links
user id of owner
group id owner
device type if inode device *
size in bytes
time of last access
time of last modification
time of last change
blocksize for filesystem I/O *
number of blocks allocated

* - only valid on systems supporting the st_blksize type - - other systems (i.e. Windows) return - 1

lstat

lstat - - ÆÄÀÏÀÌ³ª symbolic linkÀÇ stat Á¤º¸¸¦ ±¸Ñ´Ù.

Description

array lstat(string filename);

Gathers the statistics of the file or symbolic link named by filename. This function is identical to the **stat()** function except that if the *filename* parameter is a symbolic link, the status of the symbolic link is returned, not the status of the file pointed to by the symbolic link.

Returns an array with the statistics of the file with the following elements:

device
inode
number of links
user id of owner
group id owner
device type if inode device *
size in bytes
time of last access
time of last modification
time of last change
blocksize for filesystem I/O *
number of blocks allocated

* - only valid on systems supporting the st_blksize type - - other systems (i.e. Windows) return - 1

symlink

symlink - - symbolic link¸¦ ¸¸µç´Ù.

Description

int symlink(string target, string link);

symlink() creates a symbolic link from the existing *target* with the specified name *link*.

See also **link()** to create hard links, and **readlink()** along with **linkinfo()**.

tempnam

tempnam - - À¯ÀÏÇÑ ÆÄÀÏ¸íÀ» ¸¸µç´Ù.

Description

string tempnam(string dir, string prefix);

Creates a unique temporary filename in the specified directory. If the directory does not exist, **tempnam()** may generate a filename in the system's temporary directory.

Returns the new temporary filename, or the null string on failure.
Example 1. tempnam() example

```
$tmpfname = tempnam( "/tmp", "FOO" );
```

touch

touch - - ÆÄÀÏÀÇ ¼öÁ¤ ½Ã°¢À» ¼³Á¤ÇÑ´Ù.

Description

int touch(string filename, int time);

Attempts to set the modification time of the file named by filename to the value given by time. If the option time is not given, uses the present time.

If the file does not exist, it is created.

Returns true on success and false otherwise.

umask

umask - - ÇöÀçÀÇ umask¸¦ º¯°æÇÑ´Ù.

Description

int umask(int mask);

Umask() sets PHP's umask to mask & 0777 and returns the old umask. When PHP is being used as a server module, the umask is restored when each request is finished.

Umask() without arguments simply returns the current umask.

unlink

unlink - - ÆÄÀÏÀ» Áö¿î´Ù.

Description

int unlink(string filename);

Deletes *filename*. Similar to the Unix C unlink() function.

Returns 0 or FALSE on an error.

See also **rmdir()** for removing directories.

XVI. Functions related to HTTP

Table of Contents

ÀÌ ÇÔ¼öµéÀº HTTP protocol leve¿¡¼ µ¥ÀÌÅ¸¦ remote browser·Î Á÷Á¢ Àü¼ÛÇÒ ¼ö ÀÖµµ·Ï ÇØ ÁØ´Ù.

header

header - - HTTP Çì´õ µ¥ÀÌÅ¸¦ Âü¾îÁØ ±×´ë·Î º¸³½´Ù.

Description

```
int header(string string);
```

The **Header()** function is used at the top of an HTML file to send raw HTTP header strings. See the HTTP 1.1 Specification for more information on raw http headers. *Note:* Remember that the **Header()** function must be called before any actual output is sent either by normal HTML tags or from PHP. It is a very common error to read code with **include()** or with auto_prepend and have spaces or empty lines in this code that force output before **header()** is called.

```
Header("Location: http://www.php.net");  /* Redirect browser to PHP web site */
exit;  /* Make sure that code below does not get executed when we redirect. */
```

PHP scripts often generate dynamic HTML that must not be cached by the client browser or any proxy caches between the server and the client browser. Many proxies and clients can be forced to disable caching with

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");             // Date in the past
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . "GMT"); // always modified
header("Cache-Control: no-cache, must-revalidate");           // HTTP/1.1
header("Pragma: no-cache");                                   // HTTP/1.0
```

setcookie

setcookie - - cookie¸¦ ¼³Á¤ÇÏµµ·Ï µ¥ÀÌÅ¸¸¦ °ê¶ó¿ìÀú·Î °¸³½´Ù.

Description

int setcookie(string name, string value, int expire, string path, string domain, int secure);

SetCookie() defines a cookie to be sent along with the rest of the header information. All the arguments except the *name* argument are optional. If only the name argument is present, the cookie by that name will be deleted from the remote client. You may also replace any argument with an empty string (*""*) in order to skip that argument. The *expire* and *secure* arguments are integers and cannot be skipped with an empty string. Use a zero (*0*) instead. The *expire* argument is a regular Unix time integer as returned by the **time()** or **mktime()** functions. The *secure* indicates that the cookie should only be transmitted over a secure HTTPS connection. Some examples follow:
Example 1. SetCookie examples

```
SetCookie("TestCookie","Test Value");
SetCookie("TestCookie",$value,time()+3600);  /* expire in 1 hour */
SetCookie("TestCookie",$value,time()+3600,"/~rasmus/",".utoronto.ca",1);
```

Note that the value portion of the cookie will automatically be urlencoded when you send the cookie, and when it is received, it is automatically decoded and assigned to a variable by the same name as the cookie name. ie. to see the contents of our test cookie in a script, simply do:

```
echo $TestCookie;
```

For more information on cookies, see Netscape's cookie specification at http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 with Service Pack 1 applied does not correctly deal with cookies that have their path parameter set.

XVII. Hyperwave functions

(¿ªÀÚÁÖ : ÀÌ°Î°ÐÀº ¹ø¿ªÀÌ ¹«Ã´ ¼î·Á¿ü´Ù. ÀßºøµÈ ´Ü¼îµµ ¸¹Ò°í, ÀÇ¹Ì°¡ íÈ®ÇÏÁö ¼ÊÀº °Î°Ðµµ ¸¹Ò´Ù. ¶ÇÇÑ ¿ªÀÚ°¡ »ç¿ëÇØ °¸Áö ¸øÇÑ ±â´ÉÀÌ±â ¶§¹®¿¡ ±× ³»¿ëÀ» ÀßºÑ¼ËÄö ¸øÇÏ´Â ÀÌÀ¯µµ ÀÖ´Ù. ¿À¿ªÀÌ ÀÖÀ» ¼öµµ ÀÖÀ¸¸Ç·Î °¡´ÉÇÏ¸é ¿ø¹®µµ ÇÔ²² °¸µµ·Ï ÇÑ´Ù.)
Table of Contents

Introduction

Hyperwave´Â GrazÀÇ IICM¿¡¼- °³¹ßµÇ¾ú´Ù. ÀÌ°ÍÀº Ã³À½¿¡´Â Hyper - G¶ó´Â ÀÌ¸§À¸·Î ½ÃÀÛµÇ¾ú´Ù°¡ »ó÷È-µÇ¸é°Å Hyperwave¶ó´Â ÀÌ¸§À¸·Î ¹Ù²î¾ú´Ù. (¾Æ¸¶ ±â¾ï¿¡´Â 1996³âÀÎ °Í °°´Ù.)

Hyperwave´Â free software°¡ ¾Æ´Ï´Ù. ÇöÀç ¹öÀüÀº 4.0ÀÌ°í ÀÌ ¹öÀüÀº www.hyperwave.com¿¡¼- ±¸ÇÒ ¼ö ÀÖ´Ù. 30ÀÏ°£ »ç¿ëÇØ º¼ ¼ö ÀÖ´Â ½Ã°£ Á¦¾àÀÌ ÀÖ´Â ¹öÀüÀº ¹ÞÀ» ¼ö ÀÖÀ» °ÍÀÌ´Ù.

Hyperwave ´Â µ¥ÀÌÅÍ°£ÀÌ½º¿Í ºñ½ÁÇÑ information ½Ã½ºÅÛ(HIS, Hyperwave Information Server)ÀÌ´Ù. ÀÌ°ÍÀÇ ÃÊÁ¡À° ¹®¼-¸¦ °¸°üÇÏ°í °ü¸®Ç´Â °ÍÀÌ´Ù. ¿©±â¼- ¹®¼-´Â ÆÄÀÏ·Î ÀúÀåµÉ ¼ö ÀÖ´Â ¸ðµç Á¾·ùÀÇ µ¥ÀÌÅÍ¸¦ ÀÇ¹ÌÇÑ´Ù. °¢°¢ÀÇ ¹®¼-´Â ±×°ÍÀÇ object ·¹ÄÚµå¸¦ µ¿ÀÏÇÑ´Ù. ±× object ·¹ÄÚµå¿¡´Â ÇØ´ç ¹®¼-¿¡ ´ëÇÑ °¢Á¾ meta data°¡ ÀúÀåµÈ´Ù. ±× meta data´Â »ç¿ëÀÚ°¡ È®Åå°¡´ÉÇÑ ¼Ó¼º(attribute)µéÀÇ ¸ñ·Ï(list)À¸·Î µÇ¾îÀÖ´Ù. ¾î¶² ¼Ó¼ºµéÀº Hyperwave ¼-¹ö¿¡ ÀÇÇØ¼- ¹³Á¤µÈ´Ù. ´Ù¸¥ ¼Ó¼ºµéÀº »ç¿ëÀÚ°¡ ¼öÁ¤ÇØ ÁÙ ¼ö ÀÖ´Ù.

¹®¼-¿Ü¿¡µµ ¹®¼-¿¡ Æ÷ÇÔµÈ ¸ðµç hyper link±îÁöµµ object ·¹ÄÚµå·Î ÀúÀåµÈ´Ù. ¹®¼-¸¦ µ¥ÀÌÅÍ°£ÀÌ½º¿¡ ÀúÀåÇÒ ¶§ ¹®¼-¿¡ Æ÷ÇÔµÈ hyper link´Â ¹®¼-¿¡¼-´Â »èÁ¦µÇ°í °³°³ÀÇ object·Î ÀúÀåµÈ´Ù. ÇØ´ç linkÀÇ object ·¹ÄÚµå´Â ±× ¸µÅ©°¡ ¹®¼-¿¡¼-Â÷ÁöÇÏ´Â ½ÃÀÛ À§Ä¡¿Í ³¡³ª´Â À§Ä¡¿¡ ´ëÇÑ Á¤º¸¸¦ ÀúÀåÇÏ°í ÀÖ°Ô µÈ´Ù. ¿ø·¡ÀÇ ¹®¼-¸¦ ¾ò±â À§ÇØ¼- ¿©·¯°ÐÀº ¿ì¼± ¸µÅ©°¡ ¾ø´Â ¹®¼-¸¦ °Ë»öÇÑ ÈÄ¿¡ ¸µÅ©¸¦ ¸®½ºÆ®ÇÏ°í, ±×°Íµéº» ³¢¿ö ³Ö´Â´Ù. (**hw_pipedocument()**¿Í **hw_gettext()** ÇÔ¼ö°¡ ÀÌ¸¦ À§ ÇØ »ç¿ëµÈ´Ù.) ¹®¼-¿¡¼- link¸¦ °Ð¸®ÇÏ´Â °ÍÀÇ ÀåÁ¡À° ¸íÈ®ÇÏ´Ù. ¿ì¼± ¸µÅ©ÀÇ ³»¿ëÀ» ¼öÁ¤ÇÏ´Â °ÍÀÌ °£´ÜÇÏ´Ù. ¸µÅ©ÀÇ ¼öÁ¤ À¸·Î ÀüÃ¼¿¡ ¿µÇâÀ» ¹ÌÄ¡Áö ¾Ê±â ¶§¹®ÀÌ´Ù. ¶ÇÇÑ ¹®¼-¸¦ ¼öÁ¤ÇÏÁö ¾Ê°íµµ ¸µÅ©¸¦ ÃßºÐÇÒ ¼öµµ ÀÖ´Ù.

hw_pipedocument()¿Í **hw_gettext()**¸¦ »ç¿ëÇÏ¿© ¸µÅ©¸¦ »ðÀÔÇÏ´Â °ÍÀº °¸±â°¸´Ù ½¬¿îÀÏÀÌ ¾Æ´Ï´Ù. ¸µÅ©ÀÇ »ðÀÔÀº ¹®¼-ÀÇ ¸íÈ®ÇÑ °èÃþ ±¸Á¶¸¦ ¿ä±¸ÇÑ´Ù. À¥ ¼-¹ö¿¡¼- ÀÌ°ÍÀº ÆÄÀÏ ½Ã½ºÅÛÀ¸·Î ¼³Á¤ÇÑ´Ù. ±×·¯³ª Hyperwave´Â ÇØ´ç ÆÄÀÏ ½Ã½ºÅÛ °èÃþ ±¸Á¶¿Í´Â °ü°è¾øÀ´Â °íÀ¯ÀÇ °èÃþ±¸Á¶¿Í ÀÌ¸§À» °¡Áö°Ô µÈ´Ù. ±×·¯¹Ç·Î, ¸µÅ©¸¦ ¸§´Â ¿ì¼± Hyperwave °èÃþ ±¸Á¶¿Í namespace¸¦ À¥ÀÇ °èÃþ±¸Á¶¿Í namespace·Î ¸ÊÇÎÇÏ´Â °Í°ÎÅÍ ÇÏ¿©¾ß ÇÑ´Ù. Hyperwave¿Í À¥ °£ÀÇ °¡Åå ±â°»ÀûÀÎ Â÷ÀÌ´Â ÀÌ¸§ÀÇ ¸íÈ®ÇÑ ±¸°°¼°°ú Hyperwave °èÃþ±¸Á¶¿¡ ÀÖ´Ù. Hyperwave¿¡¼- ÀÌ¸§Àº object°¡ °èÃþ±¸Á¶¿¡¼-ÀÇ À§Ä¡¿¡ ´ëÇÑ Á¤º¸¸¦ °¡Áö°í ÀÖÁö ¾Ê´Ù. ±×·¯³ª À¥¿¡¼-´Â ÀÌ¸§À¸·Î °èÃþ±¸Á¶ÀÇ ¾îµð¿¡ À§Ä¡ÇÏ´Â°¡¿¡ ´ëÇÑ Á¤º¸¸¦ Æ÷ÇÔÇÏ°í ÀÖ°Ô µÈ´Ù. ÀÌ·Î¼- µÎ°¡Áö ¸íÈ®ÇÑ °¡´ÉÇÏ´Ù. Hyperwave objectÀÇ Hyperwave °èÃþ±¸Á¶¿Í ÀÌ¸§À» »ç¿ëÇÏ¿© URL¿¡ ¹Ý¿µÇÏ´Â ¹æ¹ý°ú ÀÌ¸§¸¸À» »ç¿ëÇÏ´Â ¹æ¹ýÀÌ´Ù. ÀÏ¹ÝÀûÀ¸·Î °£´ÜÇÑ ¹æ¹ýÀº ÀÌ¸§¸¸À» »ç¿ëÇÏ´Â ¹æ¹ýÀÌ´Ù. Hyperwave °èÃþ±¸Á¶ ¾È¿¡ Á¸ÀçÇÏ´Â 'my_object'ÀÌ¶ó´Â ÀÌ¸§À» °¡Áø Hyperwave object´Â 'http://host/my_object' ·Î ¸ÊÇÎµÉ´Ù. Hyperwave °èÃþ±¸Á¶¿¡¼-´Â 'parent/my_object' ¶ó´Â ÀÌ¸§À» °¡Áø object°¡ 'my_object'ÀÇ ¹Ø¿¡(child) ÀÖÀ» ¼öµµ ÀÖ´Ù. ¹°·Ð, À¥ namespace¿¡¼-´Â ±×°ÍÀÌ »ó¹ÝµÇ´Â °ÍÀÌ¸ç »ç¿ëÀÚµé¿¡°Ô È¥¶õ¾Æ °¡Á®´Ù ÁØ´Ù. ÀûÀýÇÑ obect ÀÌ¸§À» ¼±ÅÃÇÏ´Â °Í¸¸ÀÌ ÀÌ È¥¶õ¾Æ ÇÇÇÒ ¼ö ÀÖ´Ù.

ÀÌ °áÁ¤À» ³»¸®°Ô µÇ¸é µÎ°ÐÂ° ¹®Á¦°¡ »ý±ä´Ù. ¾î¶»°Ô php3¿¡ Æ÷ÇÔ½ÅÅ³ °ÍÀÎ°¡? http://host/my_objectÀÇ URL·Î´Â ¾î¶²

php3 ½ºÅ©¸³Æ®µµ È£ÃâÇÏÁö ¸øÇÑ´Ù. ÀÌ¸¦ À§ÇØ¼´Â ¿©·¯ºÐÀÌ À¥¼¹ö¿¡ ÀÌÁÖ¼Ò¸¦ 'http://host/php3_script/my_object'°°Àº ÁÖ¼Ò·Î ÀçÀ¼³Á¤ÇÏµµ·Ï ¾Ë·ÁÁÖ¾î¾ß ÇÏ°í, $PATH_INFO º¯¼ö¿Í Hyperwave serverÀÇ 'my_object'¶ó´Â ÀÌ¸§ÀÇ object¸¦ °Ë»öÇÏ¿© Æò°¡ÇÏ´Â 'php3_script'¶ó´Â ½ºÅ©¸³Æ®¡ ÀÖ¾î¾ß ÇÑ´Ù.

¿©±â¿¡´Â ½±°Ô °íÄ¥ ¼ö ÀÖ´Â °áÁ¡ÀÌ ÇÏ³ª ÀÖ´Ù. À¥¼¹ö¿¡¼- ¾ðÁ¦³ª ´Ù¸¥ ¹®¼-·ÎÀÇ Á¢±ÙÀÌ °¡´ÉÇÑ URLÀ» Àç¼³Á¤ÇÒ ¼ö ÀÖ´Â °ÍÀº ¾Æ´Ï¶ó´Â °ÍÀÌ´Ù. Hyperwave ¼-¹ö·ê °Ë»öÇÏ±â À§ÇÑ php3 ½ºÅ©¸³Æ®´Â ¾Æ¸¶ °ÒÀº¡´ÉÇÒ °ÍÀÌ´Ù. ±×·¯¹Ç·Î ¿©·¯ºÐÀº ÃÖ ¼ÒÇÑ http://host/Hyperwave·Î ½ÃÀÛÇÏ´Â °Í °°Àº Æ¯Á¤ URLÀº Á¦¿Ü½ÅÅ°´Â µÎ ¹øÂ° Àç¼³Á¤ ±ÔÄ¢À» ¼³Á¤ÇÒ ÇÊ¿ä°¡ ÀÖ´Ù. ÀÌ °ÍÀÌ À¥ÀÇ namespace¿Í Hyperwave ¼-¹ö¸¦ °øÀ¯ÇÏ´Â ±â°»ÀÌ µÈ´Ù.

¸µÅ©ÀÇ »ðÀÛÀº À§ÀÇ ¸ÞÄ«´ÏÁòÇÏ¿¡¼- ÀÌ·ç¾î Áø´Ù. ¸¸¾à php3°¡ moduleÀÌ³ª CGI ½ºÅ©¸³Æ®°¡ ¾Æ´Ñ µ¶³ ½ÇÇà ÇÁ·Î±×·¥ÀÌ ¶ó¸é(¿¹¸¦µé¾î CD-ROM¿¡ ÀÖ´Â Hyperwave ¼-¹öÀÇ ³»¿ëÀ» ´ýÇÁÇØ °»´Ù´øÄö ÇÏ´Â) ÀÌ µ¿ÀÛÀº ¸Å¸ì °¹ÀâÇØÁø´Ù. ÀÌ·± °æ¿ì¶ó¸é Hyperwave °èÅþ±¸Á¶¸¦ À¯ÁöÇÏ°í ÀÌ¸¦ ¹ÙÅÁ¸·Î ÆÄÀÏ ½Ã½ºÅÛ¿¡ ¸ÊÇÑ´Ù. ¸¸¾à Hyperwave °èÅþ±¸Á¶¸¦ ÆÄÀÏ ½Ã½º ÅÛÀÇ °èÅþ±¸Á¶¿¡ ±×´ë·Î ¹Ý¿µÇÏ°íÀÚ ÇÑ´Ù¸é °èÅþ±¸Á¶¿Í object ÀÌ§°£¿¡ ð¼øÀÌ »ý±æ ¼ö ÀÖ´Ù ('/'ÀÌ Æ÷ÇÔµÈ ÀÌ§ÀÇ °æ¿ì). µû¶ó¼- '/'´Â ¹Ýµå½Ã '_'°°Àº ´Ù¸¥ ±ÛÁÚ·Î ´ëÄ¡µÇ¾î¾ß¸¸ ÇÑ´Ù.

Hyperwave ¼-¹ö¿Í Åë½ÅÇÏ´Â ³×Æ®¿öÅ© ÇÁ·ÎÅäÄÝÀº HG-CSP (Hyper-G Client/Server Protocol)ÀÌ¶ó°í °Ò¸°´Ù. ÀÌ°ÍÀº Æ¯Á¤ µ¿ÀÛ(object ·¹ÄÚµå¸¦ ÀÏ°Å¿À´Â °Í °°Àº)À» ½ÃÀÛÇÏ´Â ¸Þ¼¼Áö¿¡ ±â¹ÝÇÑ´Ù. Hyperwave ¼-¹öÀÇ ÃÊ±â ¹öÀü¿¡´Â ¼-¹ö¿Í ÀÇ Åë¼ÅÀ» Á¦°øÇÏ±â À§ÇØ Harmony¿Í Amadeus¶ó´Â µÎ °³ÀÇ °íÀ¯ÇÑ Å¬ÍóÀÌ¾ðÆ®¡ ÀÖ¾ú´Ù. ±×·¯³ª ÀÌ µÎ°¡Áö´Â Hyperwave°¡ »ó¿ëÈ-µÇ¸é¼- »ç¶óÁö°í, wavemaster ¶ó´Â °ÍÀ¸·Î ´ëÄ¡µÇ¾ú´Ù. wavemaster´Â HTTP¿¡¼- HG-CSP·ÎÀÇ ÇÁ·ÎÅäÄÝ °¯È¯±â(protocol converter) °°Àº °ÍÀÌ´Ù. ÀÌ°ÍÀº µ¥ÀÌÅÍ°£ÀÌ½º¸¦ °ü¸®¿Í ¹®¼-ÀÇ Ç¥½Ã µî ðµç °ÍÀ» À¥ È¯°æ¿¡¼- ÇØ°áÇÏ·Á´Â »ý°¢ÀÌ´Ù. wavemaster´Â »ç¿ëÀÚ°¡ ÀÎÅÍÆäÀÌ½º¸¦ ¼öÁ¤ÇÏ´Â Æ¯Á¤ µ¿ÀÛÀ» À§ÇÑ ÀÏ·ÃÀÇ placeholderµéÀ» Á¦°øÇÑ´Ù. ÀÌ ÀÏ·ÃÀÇ placeholderµéÀ» PLACE ¾ð¾î¶ó°í °Î¸´Ù. PLACE´Â ÁøÁ¤ÇÑ ÇÁ·Î±×·¡¹Ö ¾ð¾î³ª ÀÌ¸¦ À§ÇÑ ´Ù¥ extension¿¡ °ñÇØ¼-´Â ¸¹À° ¸é¿¡¼- °ÎÁ·ÇÏ´Ù. PLACE´Â ´ÜÁö placeholder °³¼ö¸¦ ´Ã·ÁÁÖ´Â °Í »ÓÀÌ´Ù. µû¶ó¼- JavascriptÀÇ »ç¿ëÀÌ ÇÊ¼öÀûÀ¸·Î ¿ä±¸µÈ´Ù.

PHP3¿¡ Hyperwave Áö¿øÀ» Ãß°¡ÇÔ¸·Î½á, interface customisation°Î°Ð¿¡¼- PHP3ÀÇ ÇÁ·Î±×·¡¹Ö ¾ð¾îÀÇ °ÎÁ·ÇÑ °Î°ÐÀ» °¸ÃæÇÒ ¼ö ÀÖ´Ù. ÀÌ°ÍÀº HG-CSP¿¡ ÀÇÇØ Á¤ÀÇµÈ ðµç ¸Þ½ÁÁö¸¦ ±¸ÇöÇÏ´Â °ÍÀÌ°í, ³ª¾Æ°¡ ¿Ï¼ºµÈ ¹®¼-¸¦ °Ë»öÇÏ´Â µîÀÇ ´õ °-·ÂÑ ¸í·ÉéÀ» Ãß°¡Ç¿´Ù.

Hyperwave´Â Á¤°¸ÀÇ Æ¯Á¤ÇÑ ´ÜÆíµé¿¡ ÀÌ¸§À» °ùÀÌ±â À§ÇÑ ÀÚ½Å¸¸ÀÇ ÀÛ¸í(terminology)À» °¡Áö°í ÀÖ´Ù. ÀÌ°ÍÀº ±¤¹ü À§ÇÏ°Ô È®´ëµÇ°í ÀÖ´Ù. ´ë°Î°ÐÀÇ ÇÔ¼öµéÀº ´ÙÀ½ µ¥ÀÌÅÍ Å¸ÀÔ Áß¿¡ ÇÏ³ª¿¡ ÀÛ¿ëÇÑ´Ù.

object ID: Hyperwave ¼-¹ö¿¡ ÀÖ´Â °³°³ÀÇ object¿¡ ´ëÀÀÇÏ´Â À¯ÀÏÇÑ Á¤¼ö°ª. ÀÌ°ÍÀº ¶ÇÑ object ·¹ÄÚµåÀÇ ÇÑ ¼Ó¼º ÀÌ±âµµ ÇÏ´Ù. Object id´Â ¶§¶§·Î Æ¯Á¤ object¸¦ ÁöÄ¡Ï±â À§ÇÑ input ÆÄ¶ó¸ÞÅÍ·Îµµ »ç¿ëµÈ´Ù.

object record: ¼Ó¼º=°ª(attribute=value)ÀÇ ÇüÅÂ·Î Â¦Áö¿öÁø ¼Ó¼º-°ªÀÇ ¹®ÀÚ¿-. °¢°¢ÀÇ ½ÖÀº carriage return("\r") À¸·Î ±¸ºÐµÈ´Ù. ÇÏ³ªÀÇ object ·¹ÄÚµå´Â **hw_object2array()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ÇÏ³ªÀÇ object ¹è¿-·Î °¯È¯µÈ´Ù. ¸î¸îÀÇ ÇÔ¼ö´Â object ·¹ÄÚµå¸¦ °á°ú°ªÀ¸·Î µ¹·ÁÓ´Âµ¥, ÀÌ·± ÇÔ¼öµéÀÇ ÀÌ§ÀÌ obj·Î ³¡-´Ù.

object array: objectÀÇ ¸ðµç ¼Ó¼º(attribute)µéÀ» ´ã°í ÀÖ´Â ¿¬ÇÕµÈ(associated) ¹è¿-(key-value°¡ ½ÖÀ¸·Î ÀúÀåµÇ´Â ¹è¿-). ¹è¿-ÀÇ key·Î´Â ¼Ó¼ºÀÇ ÀÌ§ÀÌ »ç¿ëµÈ´Ù. ¸¸¾à ¾î¶² ¼Ó¼ºÀÌ ÇÑ object ·¹ÄÚµå¿¡ ÇÑ¹ø ÀÌ»ó ³ªÅ¸³´Ù¸é, ±×°ÍÀº ´Ù¥ ÀÎµ¦½º¸ç°Å³ª ¿¬ÇÕµÈ ¹è¿-¿¡ ÀúÀåÇÑ´Ù. titleÀÌ³ª keyword, description °ªÀº »ç¿ë ¾ð¾î¿¡ Á¾¼ÓÀûÀÎ ¼Ó¼º¸éÀº ¾ð¾îÀÇ ¾àÈ£¸¦ key·Î »ç¿ëÇÏ¿© ¿¬ÇÕ ¹è¿-À» Çü¼ÖÇÑ´Ù. ´Ù¥ ðµç °¹ÇÕÀûÀÎ ¼Ó¼ºµéÀº ÀÎµ¦½ºµÈ ¹è¿-·Î Çö¼ºµÈ´Ù. php3 ÇÔ¼öµéÀº Àý´ë·Î object array¸¦ ÇÔ¼öÀÇ °á°ú°ªÀ¸·Î µ¹·ÁÖÀ½ ¾Ê´Â´Ù.

hw_document: ÀÌ Å¸ÀÔÀº HTMLÀÌ³ª PDF °°Àº ½ÇÁ¦ ¹®¼-¸¦ °ã¾Æµî´Â ¿ÏÀüÈ÷ »õ·Î¿î Å¸ÀÔÀÌ´Ù. ÀÌ ÇüÅÂ´Â HTML ¹®¼-¿¡ ´ëÇÏ¿© ¾î´ÀÁ¤µµ ÃÖÀûÈ- µÇ¾îÖÀ¸³ª ´Ù¥ ÇöÅÂÀÇ ¹®¼-¿¡µµ »ç¿ëÇÒ ¼ö ÀÖ´Ù.

object ·¹ÄÚµåÀÇ ¹è¿-¸¦ ÝÈ¯ÇÏ´Â ¸î¸î ÇÔ¼öµéÀ» ÇØ´ç ·¹ÄÚµåµé¿¡ ´ëÇÑ Åë°èÀû Á¤º¸¸¦ °¡Áø ¿¬ÇÕ ¹è¿-µµ ÇÔ²² ¹ÝÈ¯ÇÑ´Ù. ¹ÝÈ¯µÈ ÀüÃ¼ object ·¹ÄÚµåÀÇ ¹è¿-ÀÇ ¸¶Ö¸· ¿ø¼Ò¡ ÀÌ ¹è¿-À¸´Ù. Åë°èÀû Á¤º¸¸¦ °¡Áø ¹è¿-¿¡´Â ´ÙÀ½°ú °°Àº ¿ä¼ÒµéÀÇ Á¤º¸¸¦ ÀÖ´Ù.

Hidden	PresentationHints ¼Ó¼ºÀÌ HiddenÀ¸·Î ¾³Á¤µÈ object ·¹ÄÚµåÀÇ °³¼ö (Number of object records with attribute PresentationHints set to Hidden.)
CollectionHead	PresentationHints ¼Ó¼ºÀÌ CollectionHeadÀ¸·Î ¾³Á¤µÈ object ·¹ÄÚµåÀÇ °³¼ö (Number of object records with attribute PresentationHints set to CollectionHead.)
FullCollectionHead	PresentationHints ¼Ó¼ºÀÌ FullCollectionHeadÀ¸·Î ¾³Á¤µÈ object ·¹ÄÚµåÀÇ °³¼ö (Number of object records with attribute PresentationHints set to FullCollectionHead.)
CollectionHeadNr	PresentationHints ¼Ó¼ºÀÌ CollectionHeadÀ¸·Î ¾³Á¤µÈ object ·¹ÄÚµå ¹è¿-ÀÇ index (Index in array of object records with attribute PresentationHints set to CollectionHead.)
FullCollectionHeadNr	PresentationHints ¼Ó¼ºÀÌ FullCollectionHeadÀ¸·Î ¾³Á¤µÈ object ·¹ÄÚµå ¹è¿-ÀÇ index (Index in array of object records with attribute PresentationHints set to FullCollectionHead.)
Total	object ·¹ÄÚµåÀÇ ÀüÃ¼ °³¼ö (Total number of object records.)

Apache¿ÍÀÇ ÅëÇÕ (Integration with Apache)

Hyperwave ¸ðµâÀº PHP3°¡ ¾Æ¾ÄÄ¡ ¸ðµâ·Î ÄÄÆÄÀÏµÇ¾úÀ» ¶§ °¡Àå Àß »ç¿ëµÉ ¼ö ÀÖ´Ù. ÀÌ °æ¿ì¿¡ ¾Æ¾ÄÄ¡°¡ ÀÚ½ÅÀÇ Àç ÀÛ¼º(rewriting) ¿£Áø¿Í »ç¿ëÇÏ°Ô µÇ¸é Hyperwave ¼-¹öÀÇ Á¸Àç´Â »ç¿ëÀÚ°¡ °ÅÀÇ ´Á³¥ ¼ö ¾ø°Ô µÈ´Ù. ´ÙÀ½ÀÇ ³»¿ëµéÀÌ ÀÌ °ÍÀ» ¼³¸íÇØ ÁÙ °ÍÀÌ´Ù.

Hyperwave¸¦ Áö¿øÇÏ´Â ¾Æ¾ÄÄ¡·Î ¸¸µé¾îÁø PHP3°¡ wavemaster±â¹ÝÀÇ µ¶ÀÚÀûÀÎ Hyperwave ¼Ö·ç¼ÇÀ» ´ë½ÅÇÏ·Á ÇÒ ¶§ °ÎÅÍ, ¾Æ¾ÄÄ¡ ¼-¹ö°¡ Hyperwave À¥ ÀÎÅÍÆäÀÌ½º¸¦ Áö¿øÇÏ´Â À¯ÀÍÇÑ ¼-¹ö°¡ µÇ¸®¶ó ´ç¿¬È÷ ÃßÁ¤µÇ¾ú´Ù.

ÀÌ°ÍÀº ¹Ýµå½Ã ÇÊ¿äÇÑ °ÍÀº ¾Æ´ÏÁö¸¸ ¼³Á¤À» ½±°Ô ÇØÁØ´Ù. °³³äÀº ´Ü¼øÇÏ´Ù. ¿ì¼±, ¿©·¯°ÐÀº PATH_INFO º¯¼ö¸¦ Æò°¡ÇÏ°í ÀÌ º¯¼öÀÇ °ªÀ» Hyperwave object·Î ´Ù·ê PHP3 ½ºÅ©¸³Æ®°¡ ÇÊ¿äÇÏ´Ù. ÀÌÁ¦ ÀÌ ½ºÅ©¸³Æ®ÀÇ ÀÌ¸§À» 'Hyperwave'¶ó°í ºÎ¸£ÀÚ. http://your.hostname/Hyperwave/name_of_object¶ó´Â URLÀº ¾Æ¸¶µµ 'name_of_object'¶ó´Â ÀÌ¸§À» °¡Áø Hyperwave object¸¦ ¹ÝÈ¯ÇÒ °ÍÀÌ´Ù. ½ºÅ©¸³Æ®´Â ±× objectÀÇ Å¸ÀÔ¿¡ µû¶ó¼- ÀûÀýÈ÷ ¹ÝÀÀÇÑ´Ù. ¸¸¾à Å¸ÀÔÀÌ collectionÀÌ¶ó¸é ¾Æ¸¶µµ childrenÀÇ ¸®½ºÆ®¸¦ ¹ÝÈ¯ÇÑ´Ù. ¸¸¾à Å¸ÀÔÀÌ ¹®¼-(document)¶ó¸é mime Å¸ÀÔºú ±× ³»¿ëÀ» ¹ÝÈ¯ÇÑ´Ù. ¾Æ¾ÄÄ¡ ÀçÀÛ¼º ¿£Áø(apache rewriting engine)ÀÌ »ç¿ëµÈ´Ù¸é ¾à°£ÀÇ °³¼±À» ¼öÀ» ¼ö ÀÖ´Ù. http://your.hostname/name_of_object°°Àº ÇüÅÂ·Î URLÀ» »ç¿ëÇÏ¿© object¸¦ ¹ÝÈ¯ÇÑ´Ù¸é »ç¿ëÀÚ ÀÛÀå¿¡¼- Á»´õ Á÷°üÀûÀÌ ¼ö ÀÖ´Ù. ÀÌ¸¦ À§ÇÑ ÀçÀÛ¼º(rewriting) ±ÔÄ¢Àº ´Ù À½°ú °°ÀÌ ¸Å¿ì ½±´Ù. :

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

ÀÌÁ¦ ¸ðµç URLÀº Hyperwave ¼-¹öÀÇ object¿Í ¿¬°áµÈ´Ù. ÀÌ°ÍÀº °ñ±³Àû °£´ÜÈ÷ ÇØ°áÇÒ ¼ö ÀÖ´Â ÇÑ°¡Áö ¹®Á¦¸¸ ³²´Ù. ÀÌ·Î¼á 'Hyperwave' ½ºÅ©¸³Æ® ÀÌ¿ÜÀÇ ´Ù¸¥ ½ºÅ©¸³Æ®(¿¹¸¦µé¾î, °Ë»öÀ» À§ÇÑ ½ºÅ©¸³Æ® µî)´Â ½ÇÇàÇÒ ¼ö ¾ø°Ô µÈ´Ù. ÀÌ¸¦ ÇØ °áÇÏ±â À§ÇÑ ¶Ç´Ù¸¥ ÀÛ¼º(rewriting) ±ÔÄ¢Àº ´ÙÀ½°ú °°ÀÌ ÀÛ¼ºÇØ ÁØ´Ù. ¤Ó

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

ÀÌ°ÍÀ¸·Î Ãß°¡ÀûÀÎ ½ºÅ©¸³Æ®¿Í ÆÄÀÏµéÀ» À§ÇÑ /usr/local/apache/htdocs/hw µð·º¸ä¸®¸¦ ³²°ÜµÎ°Ô µÈ´Ù. ±×¸®°í ÀÌ ±ÔÄ¢Àº À§ÀÇ ±ÔÄ¢º¸´Ù ¸ÕÀú Æò°¡µÇµµ·Ï ÇÑ´Ù. ¾à°£ÀÇ °áÁ¡ÀÌ ÀÖ´Â¶¥, 'hw/'·Î ½ÃÀÛÇÏ´Â ÀÌ¸§À» °¡Áø Hyperwave object´Â Á¢±ÙÇÒ ¼ø °Ô µÈ´Ù. ±×·¯¹Ç·Î ¿©·¯°ÐÀº ±×·± ÀÌ¸§À° ¾²Áö ¾Êµµ·Ï ÇÑ´Ù. ¸¸¾à ¿©·¯°ÐÀÌ ´õ ¸¹Àº µð·º¸ä¸®(¿¹¸¦µé¾î, ÀÌ¹ÌÁö¸¦ À§ÇÑ µð·º¸ä¸®)°¡ ÇÊ¿äÇÏ´Ù¸é, »õ·Î¿î ±ÔÄ¢À» Ã·°¡ÇÏ°Å³ª ¿øÇÏ´Â µð·º¸ä¸®µéÀ» ¸ðµÎ ÇÑ µð·º¸ä¸® ¹Ø¿¡ ³õÀ¸¸é µÈ´Ù. ¸¶Áö¸·Å¸·Î, ´ÙÀ½°ú °°ÀÌÇÏ¿© ÀçÀÛ¼º ¿£Áø(rewriting engine)À» ÄÑµµ·Ï ÇÑ´Ù. :

```
RewriteEngine on
```

³» °æÇèÀ¸·Î °¸¾ÒÀ» ¶§ ¿©·¯°ÐÀ» ´ÙÀ½ÀÇ ½ºÅ©¸³Æ®µéÀÌ ÇÊ¿äÇÏ°Ô µÉ °ÍÀÌ´Ù. :

object ÀÚ½ÅÀ» µ¹·ÁÄÖ±â (to return the object itself)

°Ë»ö °¡´ÉÇÏ°Ô ÇÏ±â (to allow searching)

ÀÚ±â ÀÚ½Å¿¡ ´ëÇÑ È®ÀÎ (to identify yourself)

profile ¼³Á¤ (to set your profile)

object ¼Ó¼ºÀ» °¸¿©ÁÖ°Å³ª, »ç¿ëÀÚÀÇ Á¤º¸¸¦ °¸¿©ÁÖ°Å³ª, ¼-¹öÀÇ »óÅÂ¸¦ °¸¿©ÁÖ°Í °°Àº Á¾·ùÀÇ ÇÔ¼öµé

Todo

¾ÆÁ÷ ÇØ¾ßÇÒ ÀÏÀº ´ÙÀ½°ú °°´Ù. :

hw_InsertDocument¸¦ **hw_InsertObject()**¿Í **hw_PutDocument()**ÀÇ µÎ ÇÔ¼ö·Î ³²´©©±â

°³³äÀÇ ÇÔ¼öµéÀº ±× ÀÌ¸§À» ¾ÆÁ÷ È®Á¤µÇÁö ¾Ê¾Ò´Ù.

´ë°ÎºÐÀÇ ÇÔ¼ö´Â Ã¹ ¹øÂ° ÆÄ¶ó¸¸Å¸Í·Î ÇöÀçÀÇ ¿¬°á(connection)À» ¿ä±¸Çß´Ù. ÀÌ°ÍÀº ÇÑ °³ÀÇ ¿¬á¸¸ ÀÖ´Â °æ¿ì ´ë°ÎºÐ °° ÇÊ¿ä¾ø´Â Å¸ÀÌÇÎÀÌ ÇÊ¿äÇÏ°Ô µÈ´Ù. ÀÌ¸¦ À§ÇØ ±âº» ¿¬°á(default connection)ÀÌ ÇÊ¿äÇÒ °ÍÀÌ´Ù.

---

hw_Modifyobject

hw_Modifyobject - - object ·¹ÄÚµå¸¦ ¼öÁ¤ÇÑ´Ù.

Description

int hw_modifyobject(int connection, int object_to_change, array remove, array add);

This command allows to remove, add, or modify individual attributes of an object record. The object is specified by the Object ID *object_to_change*. The first array *remove* is a list of attributes to remove. The second array *add* is a list of attributes to add. In order to modify an attribute one will have to remove the old one and add a new one.

hw_modifyobject() will always remove the attributes before it adds attributes. The keys of both arrays are the attributes name. The value of each array element can either be an array of a string. If it is an array each attribute value is constructed by the key of each element plus a colon an the value of each element. If it is a string it is taken as the attribute value.

If you would like to change the attribute 'Name' with the current value 'books' into 'articles' you will have to create two arrays and call **hw_modifyobject()**.
Example 1. modifying an attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => "books");
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

In order to delete/add a name=value pair from/to the object record just pass the remove/add array and set the last/third parameter to an empty array.

Note: Multilingual attributes, e.g. 'Title', can be modified in two ways. Either by providing the attributes value in its native form 'language':'title' or by providing an array with elements for each language as described above. The above example would than be:
Example 2. modifying Title attribute

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or
Example 3. modifying Title attribute

```
$remarr = array("Title" => array("en" => "Books"));
$addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This removes the english title 'Books' and adds the english title 'Articles' and the german title 'Artikel'.
Example 4. removing attribute

```
$remarr = array("Title" => "");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This will remove all attributes with the name 'Title' and adds a new 'Title' attribute. This comes in handy if you want to remove attributes recursively.

If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.

Note: Only the attributes 'Title', 'Description' and 'Keyword' will properly handle the language prefix. If those attributes don't carry a language prefix, the prefix 'xx' will be assigned.

Returns TRUE if no error occurs otherwise FALSE.

hw_Children

hw_Children - - childrenÀÇ object idµé

Description

`array hw_children`(int connection, int objectID);

Returns an array of object ids. Each id belongs to a child of the collection with ID *`objectID`*. The array contains all children both documents and collections.

hw_ChildrenObj

hw_ChildrenObj - - childrenÀÇ object recordµé

Description

`array hw_childrenobj`(int connection, int objectID);

Returns an array of object records. Each object record belongs to a child of the collection with ID *`objectID`*. The array contains all children both documents and collections.

hw_Close

hw_Close - - Hyperwave connectionÀ» ´Ý´Â´Ù.

Description

int hw_close(int connection);

Returns false if connection is not a valid connection index, otherwise true. Closes down the connection to a Hyperwave server with the given connection index.

hw_Connect

hw_Connect - - Hyperwave connectionÀ» ¿¬´Ù.

Description

int hw_connect(string host, int port, string username, string password);

Opens a connection to a Hyperwave server and returns a connection index on success, or false if the connection could not be made. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple connections open at once. Keep in mind, that the password is not encrypted.

See also **hw_pConnect()**.

hw_Cp

hw_Cp - - ÁöÁ¤ÇÑ objectµéÀ» º¹»çÇÑ´Ù.

Description

int hw_cp(int connection, array object_id_array, int destination id);

Copies the objects with object ids as specified in the second parameter to the collection with the id *destination id*.

The value return is the number of copied objects.

See also **hw_mv()**.

hw_Deleteobject

hw_Deleteobject - - Æ¯Á¤ object¸¦ »èÁ¦ÇÑ´Ù.

Description

int hw_deleteobject(int connection, int object_to_delete);

Deletes the the object with the given object id in the second parameter. It will delete all instances of the object.

Returns TRUE if no error occurs otherwise FALSE.

See also **hw_mv()**.

hw_DocByAnchor

hw_DocByAnchor - - ÁöÁ¤µÈ anchor¿¡ ¸ÓÇÏ´Â objectÀÇ object id

Description

int hw_docbyanchor(int connection, int anchorID);

Returns an th object id of the document to which *anchorID* belongs.

hw_DocByAnchorObj

hw_DocByAnchorObj - - ÁöÁ¤µÈ anchor¿¡ ¼ÓÇÏ´Â objectÀÇ object record

Description

string hw_docbyanchorobj(int connection, int anchorID);

Returns an th object record of the document to which *anchorID* belongs.

hw_DocumentAttributes

hw_DocumentAttributes - - hw_documentÀÇ object record

Description

string hw_documentattributes(int hw_document);

Returns the object record of the document.

See also **hw_DocumentBodyTag()**, **hw_DocumentSize()**.

hw_DocumentBodyTag

hw_DocumentBodyTag - - hw_documentÀÇ body tagÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

string hw_documentbodytag(int hw_document);

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also **hw_DocumentAttributes()**, **hw_DocumentSize()**.

hw_DocumentContent

hw_DocumentContent - - hw_documentÀÇ ³»¿ëÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

string hw_documentcontent(int hw_document);

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also **hw_DocumentAttributes()**, **hw_DocumentSize()**, **hw_DocumentSetContent()**.

hw_DocumentSetContent

hw_DocumentSetContent - - hw_documentÀÇ ³»¿ëÀ» ¼³Á¤ÇÏ°Å³ª ÀçÁöÁ¤ÇÑ´Ù.

Description

string hw_documentsetcontent(int hw_document, string content);

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also **hw_DocumentAttributes()**, **hw_DocumentSize()**, **hw_DocumentContent()**.

hw_DocumentSize

hw_DocumentSize - - hw_documentÀÇ Å©±â

Description

int hw_documentsize(int hw_document);

Returns the size in bytes of the document.

See also **hw_DocumentBodyTag()**, **hw_DocumentAttributes()**.

hw_ErrorMsg

hw_ErrorMsg - - error message¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

string hw_errormsg(int connection);

Returns a string containing the last error message or 'No Error'. If false is returned, this function failed. The message relates to the last command.

hw_EditText

hw_EditText - - text ¹®¼¸¦ ¿ø·¡´ë·Î °¹±¸(retrieve)ÇÑ´Ù.

Description

int hw_edittext(int connection, int hw_document);

Uploads the text document to the server. The object record of the document may not be modified while the document is edited. This function will only works for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**, **hw_GetText()**.

hw_Error

hw_Error - - error ¹øÈ£¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int hw_error(int connection);

Returns the last error number. If the return value is 0 no error has occurred. The error relates to the last command.

hw_Free_Document

hw_Free_Document - - hw_document°¡ Á¡À¯ÇÏ°í ÀÖ´Â ÀÚ¿øµéÀ» Ç®¾îÁØ´Ù.

Description

int hw_free_document(int hw_document);

Frees the memory occupied by the Hyperwave document.

hw_GetParents

hw_GetParents - - parentµéÀÇ object idµé

Description

array hw_getparentsobj(int connection, int objectID);

Returns an indexed array of object ids. Each object id belongs to a parent of the object with ID *objectID*.

hw_GetParentsObj

hw_GetParentsObj - - parentµéÀÇ object recordµé

Description

array hw_getparentsobj(int connection, int objectID);

Returns an indexed array of object records plus an associated array with statistical information about the object records. The associated array is the last entry of the returned array. Each object record belongs to a parent of the object with ID *objectID*.

hw_GetChildColl

hw_GetChildColl - - child collectionµéÀÇ object idµé

Description

array hw_getchildcoll(int connection, int objectID);

Returns an array of object ids. Each object ID belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also **hw_GetChildren()**, **hw_GetChildDocColl()**.

hw_GetChildCollObj

hw_GetChildCollObj - - child collectionµéÀÇ object recordµé

Description

array hw_getchildcollobj(int connection, int objectID);

Returns an array of object records. Each object records belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also **hw_ChildrenObj()**, **hw_GetChildDocCollObj()**.

hw_GetSrcByDestObj

hw_GetSrcByDestObj - - ÁöÁ¤µÈ object¸¦ Áö½ÃÇÏ´Â anchorµéÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

array hw_getsrcbydestobj(int connection, int objectID);

Returns the object records of all anchors pointing to the object with ID *objectID*. The object can either be a document or an anchor of type destination.

See also **hw_GetAnchors()**.

hw_GetObject

hw_GetObject - - ÁöÁ¤ÇÑ object record¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

array hw_getobject(int connection, int objectID);

Returns the object record for the object with ID *objectID*.

See also **hw_GetAndLock()**.

hw_GetAndLock

hw_GetAndLock - - object record¸¦ ¹ÝÈ¯ÇÏ°í ÇØ´ç object¸¦ lockÇÑ´Ù.

Description

string hw_getandlock(int connection, int objectID);

Returns the object record for the object with ID *objectID*. It will also lock the object, so other users cannot access it

until it is unlocked.

See also **hw_Unlock()**, **hw_GetObject()**.

hw_GetText

hw_GetText - - text ¹®¼-¸¦ ¿ø·¡´ë·Î °¹±¸(retrieve)ÇÑ´Ù.

Description

int hw_gettext(int connection, int objectID, int rootID);

Returns the document with object ID *objectID*. If the document has anchors which can be inserted, they will be inserted already. The optional parameter *rootID* determines how links are inserted into the document. The default is 0 and will result in links that are constructed from the name of the link's destination object. This is useful for web applications. If a link points to an object with name 'internet_movie' the HTML link will be <A HREF="/internet_movie">. The actual location of the source and destination object in the document hierachy is disregarded. You will have to set up your web browser, to rewrite that URL to for example '/my_script.php3/internet_movie'. 'my_script.php3' will have to evaluate $PATH_INFO and retrieve the document.

If *rootID* is unequal to 0 the link is constructed from all the names starting at the object with the id *rootID* separated by a slash relative to the current object. If for example the above document 'internet_movie' is located at 'a - b - c - internet_movie' with ' - ' being the seperator between hierachy levels and the source document is located at 'a - b - d - source' the resulting HTML link would be: <A HREF="../c/internet_movie">. This is useful if you want to download the whole server content onto disk and map the document hierachy onto the file system.

This function will only work for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**.

hw_GetObjectByQuery

hw_GetObjectByQuery - - object¸¦ Ã£´Â´Ù.

Description

array hw_getobjectbyquery(int connection, string query, int max_hits);

Searches for objects on the whole server and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

See also **hw_GetObjectByQueryObj()**.

hw_GetObjectByQueryObj

hw_GetObjectByQueryObj - - object¸¦ Ã£´Â´Ù.

Description

array hw_getobjectbyqueryobj(int connection, string query, int max_hits);

Searches for objects on the whole server and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

See also **hw_GetObjectByQuery()**.

hw_GetObjectByQueryColl

hw_GetObjectByQueryColl - - collection ¼È¿¡¼- objectÀ» Ã£´Â´Ù.

Description

array hw_getobjectbyquerycoll(int connection, int objectID, string query, int max_hits);

Searches for objects in collection with ID *objectID* and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

See also **hw_GetObjectByQueryCollObj()**.

hw_GetObjectByQueryCollObj

hw_GetObjectByQueryCollObj - - collection ¾È¿¡¼- objectÀ» Ã£´Â´Ù.

Description

array hw_getobjectbyquerycollobj(int connection, int objectID, string query, int max_hits);

Searches for objects in collection with ID *objectID* and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

See also **hw_GetObjectByQueryColl()**.

hw_GetChildDocColl

hw_GetChildDocColl - - collection ¾ÈÀÇ child ¹®¼-µéÀÇ object id¸¦ ±¸Ñ´Ù.

Description

array hw_getchilddoccoll(int connection, int objectID);

Returns array of object ids for child documents of a collection.

See also **hw_GetChildren()**, **hw_GetChildColl()**.

hw_GetChildDocCollObj

hw_GetChildDocCollObj - - collection ¾ÈÀÇ child ¹®¼-µéÀÇ object record¸¦ ±¸Ñ´Ù.

Description

array hw_getchilddoccollobj(int connection, int objectID);

Returns an array of object records for child documents of a collection.

See also **hw_ChildrenObj()**, **hw_GetChildCollObj()**.

hw_GetAnchors

hw_GetAnchors - - ¹®¼-ÀÇ anchorµéÀÇ object id

Description

array hw_getanchors(int connection, int objectID);

Returns an array of object ids with anchors of the document with object ID *objectID*.

hw_GetAnchorsObj

hw_GetAnchorsObj - - ¹®¼-ÀÇ anchorµéÀÇ object record

Description

array hw_getanchorsobj(int connection, int objectID);

Returns an array of object records with anchors of the document with object ID *objectID*.

hw_Mv

hw_Mv - - objectµéÀ» ¿Å±ä´Ù.

Description

int hw_mv(int connection, array object id array, int source id, int destination id);

Moves the objects with object ids as specified in the second parameter from the collection with id *source id* to the collection with the id *destination id*. If the source id is 0 the objects will be unlinked from the source collection. If this is the last instance of that object it will be deleted.

The value return is the number of moved objects.

See also **hw_cp()**, **hw_deleteobject()**.

hw_Identify

hw_Identify - - »ç¿ëÀÚ¸¦ identifyÇÑ´Ù.

Description

`int hw_identify`(string username, string password);

Identifies as user with *username* and *password*. Identification is only valid for the current session. I do not thing this function will be needed very often. In most cases it will be easier to identify with the opening of the connection.

See also **hw_Connect()**.

hw_InCollections

hw_InCollections - - object idÀÇ object°¡ collection¿¡ ¼ÓÇÏ´Â°¡ °Ë»çÇÑ´Ù.

Description

`array hw_incollections`(int connection, array object_id_array, array collection_id array, int return_collections);

Checks whether a set of objects (documents or collections) specified by the *object_id_array* is part of the collections defined by collection *id_array*. When the fourth parameter *return_collections*is 0, the subset of object ids that is part of the collections (i.e., the documents or collections that are children of one or more collections of collection ids or their subcollections, recursively) is returned as an array. When the fourth parameter is 1, however, the set of collections that have one or more objects of this subset as children are returned as an array. This option allows a client to, e.g., highlight the part of the collection hierarchy that contains the matches of a previous query, in a graphical overview.

hw_Info

hw_Info - - connection¿¡ ´ëÇÑ Á¤º¸

Description

`string hw_info`(int connection);

Returns information about the current connection. The returned string has the following format: <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl

hw_InsColl - - collectionÀ» »ðÀÔÇÑ´Ù.

Description

`int hw_inscoll`(int connection, int objectID, array object_array);

Inserts a new collection with attributes as in *object_array* into collection with object ID *objectID*.

hw_InsDoc

hw_InsDoc - - document¸¦ »ðÀÔÇÑ´Ù.

Description

`int hw_insdoc`(int connection, int parentID, string object_record, string text);

Inserts a new document with attributes as in *object_record* into collection with object ID *parentID*. This function inserts either an object record only or an object record and a pure ascii text in *text* if *text* is given. If you want to insert a

general document of any kind use **hw_insertdocument()** instead.

See also **hw_InsertDocument()**, **hw_InsColl()**.

hw_InsertDocument

hw_InsertDocument - - ¾î¶² document¸¦ collectionÀ¸·Î upload ÇÑ´Ù.

Description

`int hw_putdocument`(int connection, int parent_id, int hw_document);

Uploads a document into the collection with *parent_id*. The document has to be created before with **hw_NewDocument()**. Make sure that the object record of the new document contains at least the attributes: Type, DocumentType, Title and Name. Possibly you also want to set the MimeType.

See also **hw_PipeDocument()**.

hw_InsertObject

hw_InsertObject - - object record¸¦ Ãß°¡ÇÑ´Ù.

Description

`int hw_insertobject`(int connection, string object rec, string parameter);

Inserts an object into the server. The object can be any valid hyperwave object. See the HG-CSP documentation for a detailed information on how the parameters have to be.

Note: If you want to insert an Anchor, the attribute Position has always been set either to a start/end value or to 'invisible'. Invisible positions are needed if the annotation has no correspondig link in the annotation text.

See also **hw_PipeDocument()**, **hw_InsertDocument()**, **hw_InsDoc()**, **hw_InsColl()**.

hw_New_Document

hw_New_Document - - »õ document¸¦ ¸¸µç´Ù.

Description

`int hw_new_document`(string document_data, string object_record, int document_size);

Returns a new Hyperwave document with document data set to *document_data* and object record set to *object_record*. The length of the *document_data* has to passed in *document_size*This function does not insert the document into the Hyperwave server.

See also **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**, **hw_InsertDocument()**.

hw_Objrec2Array

hw_Objrec2Array - - object recordÀÇ ¼Ó¼º(attributes)À» object array·Î º¯È¯ÇÑ´Ù.

Description

`array hw_objrec2array`(string object_record);

Converts an *object_record* into an object array.

hw_OutputDocument

hw_OutputDocument - - hw_document¸¦ printÇÑ´Ù.

Description

`int hw_outputdocument`(int hw_document);

Prints the document without the BODY tag.

hw_pConnect

hw_pConnect - - ¿µ±¸ÀûÀÎ(persistent) database connectionÀ» ¸¸µç´Ù.

Description

int hw_pconnect(string host, int port, string username, string password);

Returns a connection index on success, or false if the connection could not be made. Opens a persistent connection to a Hyperwave server. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple persistent connections open at once.

See also **hw_Connect()**.

hw_PipeDocument

hw_PipeDocument - - document¸¦ ¿ø·¡´ë·Î °¹±¸(retrieve)ÇÑ´Ù.

Description

int hw_pipedocument(int connection, int objectID);

Returns the Hyperwave document with object ID *objectID*. If the document has anchors which can be inserted, they will have been inserted already. The document will be transfered via a special data connection which does not block the control connection.

See also **hw_GetText()** for more on link insertion, **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**.

hw_Root

hw_Root - - root objectÀÇ object id

Description

int hw_root();

Returns the object ID of the hyperroot collection. Currently this is always 0. The child collection of the hyperroot is the root collection of the connected server.

hw_Unlock

hw_Unlock - - object¸¦ unlock ÇÑ´Ù.

Description

int hw_unlock(int connection, int objectID);

Unlocks a document, so other users regain access.

See also **hw_GetAndLock()**.

hw_Username

hw_Username - - ÇöÀç log in ÇÏ°í ÀÖ´Â userÀÇ ÀÌ¸§

Description

string hw_getusername(int connection);

Returns the username of the connection.

XVIII. Image functions

Table of Contents

¿©·¯ºÐÀº ¿©±âÀÇ ÇÔ¼öµé·Î JPEG, GIF, PNG ÀÌ¹ÌÁöÀÇ Å©±â¸¦ ¾Ë¾Æ³¾ ¼ö ÀÖ´Ù. ¶ÇÇÑ ¿©·¯ºÐÀÌ GD library(http://www.boutell.com/gd/¿¡ ÀÖ´Ù.)¸¦ °¡Áö°í ÀÖ´Ù¸é, ¿©·¯ºÐÀº GIF ÀÌ¹ÌÁö¸¦ ¸¸µé°Å³ª ¼öÁ¤ÇÒ ¼ö ÀÖ´Ù.

GetImageSize

GetImageSize - - GIF³ª JPG, PNG ±×¸²ÀÇ Å©±â¸¦ ±¸Ñ´Ù.

Description

`array getimagesize(string filename, array [imageinfo]);`

The **GetImageSize()** function will determine the size of any GIF, JPG or PNG image file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML IMG tag.

Returns an array with 4 elements. Index 0 contains the width of the image in pixels. Index 1 contains the height. Index 2 a flag indicating the type of the image. 1 = GIF, 2 = JPG, 3 = PNG. Index 3 is a text string with the correct "height=xxx width=xxx" string that can be used directly in an IMG tag.

Example 1. GetImageSize

```
<?php $size = GetImageSize("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>>
```

The optional *imageinfo* parameter allows you to extract some extended information from the image file. Currently this will return the diffrent JPG APP markers in an associative Array. Some Programs use these APP markers to embedd text information in images. A very common one in to embed IPTC http://www.xe.net/iptc/ information in the APP13 marker. You can use the **iptcparse()** function to parse the binary APP13 marker into something readable.

Example 2. GetImageSize returning IPTC

```
<?php
    $size = GetImageSize("testimg.jpg",&$info);
    if (isset($info["APP13"])) {
        $iptc = iptcparse($info["APP13"]);
```

```
        var_dump($iptc);
    }
?>
```

Note: This function does not require the GD image library.

ImageArc

ImageArc - - Å¸¿ø È£¸¦ ±× ¸° ´Ù.

Description

int imagearc(int im, int cx, int cy, int w, int h, int s, int e, int col);

ImageArc draws a partial ellipse centered at cx, cy (top left is 0,0) in the image represented by im. w and h specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the s and e arguments.

ImageChar

ImageChar - - ÇÑ ±ÛÀÚ¸¦ ¼öÆòÀ¸·Î ±×¸°´Ù.

Description

int imagechar(int im, int font, int x, int y, string c, int col);

ImageChar draws the first character of c in the image identified by id at coordinates x, y (top left is 0,0) with the color col. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

ImageCharUp

ImageCharUp - - ÇÑ ±ÛÀÚ¸¦ ¼öÁ÷À¸·Î ±×¸°´Ù.

Description

int imagecharup(int im, int font, int x, int y, string c, int col);

ImageCharUp draws the character c vertically in the image identified by im at coordinates x, y (top left is 0, 0) with the color col. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

ImageColorAllocate

ImageColorAllocate - - ±×¸²À» À§ÇÑ »ö±òÀ» ÇÒ´çÇÑ´Ù.

Description

int imagecolorallocate(int im, int red, int green, int blue);

ImageColorAllocate returns a color identifier representing the color composed of the given RGB components. The im argument is the return from the **imagecreate()** function. ImageColorAllocate must be called to create each color that is to be used in the image represented by im.

```
$white = ImageColorAllocate($im, 255,255,255);
$black = ImageColorAllocate($im, 0,0,0);
```

ImageColorTransparent

ImageColorTransparent - - Åõ¸íÇÑ »ö±òÀ» ÁöÁ¤ÇÑ´Ù.

Description

int imagecolortransparent(int im, int [col]);

ImageColorTransparent sets the transparent color in the im image to col. im is the image identifier returned by

imagecreate() and col is a color identifier returned by **imagecolorallocate()**.

The identifier of the new (or current, if none is specified) transparent color is returned.

ImageCopyResized

ImageCopyResized - - ±×¸²ÀÇ ÀÏºÎºĐÀ» º¹»çÇÏ°í Å©±â¸¦ ¹Ù²Û´Ù.

Description

int imagecopyresized(int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH);

ImageCopyResized copies a rectangular portion of one image to another image. *dst_im* is the destination image, *src_im* is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

ImageCreate

ImageCreate - - »õ image¸¦ ¸¸µç´Ù.

Description

int imagecreate(int x_size, int y_size);

ImageCreate returns an image identifier representing a blank image of size x_size by y_size.

ImageCreateFromGif

ImageCreateFromGif - - ÆÄÀÏÀÌ³ª URL·ÎºÎÅÍ »õ image¸¦ ¸¸µç´Ù.

Description

int imagecreatefromgif(string filename);

ImageCreateFromGif returns an image identifier representing the image obtained from the given filename.

ImageDashedLine

ImageDashedLine - - Á¡¼±À» ±×¸°´Ù.

Description

int imagedashedline(int im, int x1, int y1, int x2, int y2, int col);

ImageLine draws a dashed line from x1,y1 to x2,y2 (top left is 0,0) in image im of color col.

See also **imageline()**.

ImageDestroy

ImageDestroy - - image¸¦ ¸ø¾Ø´Ù.

Description

int imagedestroy(int im);

ImageDestroy frees any memory associated with image im. im is the image identifier returned by the **imagecreate()** function.

ImageFill

ImageFill - - »öÀ» Ã¤¿î´Ù.

Description

int imagefill(int im, int x, int y, int col);

ImageFill performs a flood fill starting at coordinate x, y (top left is 0,0) with color col in the image im.

ImageFilledPolygon

ImageFilledPolygon - - »öÀÌ Ã¤¿öÁø ´Ù°¢ÇüÀ» ±×¸°´Ù.

Description

int imagefilledpolygon(int im, array points, int num_points, int col);

ImageFilledPolygon creates a filled polygon in image im. points is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. num_points is the total number of vertices.

ImageFilledRectangle

ImageFilledRectangle - - »öÀÌ Ã¤¿öÁø Á÷»ç°¢ÇüÀ» ±×¸°´Ù.

Description

int imagefilledrectangle(int im, int x1, int y1, int x2, int y2, int col);

ImageFilledRectangle creates a filled rectangle of color col in image im starting at upper left coordinates x1, y1 and ending at bottom right coordinates x2, y2. 0, 0 is the top left corner of the image.

ImageFillToBorder

ImageFillToBorder - - ÁöÁ¤µÈ »öÀ¸·Î Ã¤¿î´Ù.

Description

int imagefilltoborder(int im, int x, int y, int border, int col);

ImageFillToBorder performs a flood fill whose border color is defined by border. The starting point for the fill is x,y (top left is 0,0) and the region is filled with color col.

ImageFontHeight

ImageFontHeight - - ÆùÆ®ÀÇ ³ôÀÌ¸¦ ±¸ÇÑ´Ù.

Description

int imagefontheight(int font);

Returns the pixel width of a character in font.

See also **imagefontwidth()** and **imageloadfont()**.

ImageFontWidth

ImageFontWidth - - ÆùÆ®ÀÇ ³ÐÀÌ¸¦ ±¸ÇÑ´Ù.

Description

int imagefontwidth(int font);

Returns the pixel width of a character in font.

See also **imagefontheight()** and **imageloadfont()**.

ImageGif

ImageGif - - browser³ª ÆÄÀÏ·Î ±×¸²À» Ãâ·ÂÇÑ´Ù.

Description

int imagegif(int im, string filename);

ImageGif creates the GIF file in filename from the image im. The im argument is the return from the **imagecreate()** function.

The image format will be GIF87a unless the image has been made transparent with **imagecolortransparent()**, in which case the image format will be GIF89a.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/gif content-type using the header function, you can create a PHP script that outputs GIF images directly.

ImageInterlace

ImageInterlace - - ±×¸²ÀÌ ¼¯ÀÌ°Ô ÇÏ´Â interlace ¼³Á¤À» ÄÑ°Å³ª ²ö´Ù.

Description

int imageinterlace(int im, int [interlace]);

ImageInterlace() turns the interlace bit on or off. If interlace is 1 the im image will be interlaced, and if interlace is 0 the interlace bit is turned off.

This functions returns whether the interlace bit is set for the image.

ImageLine

ImageLine - - ¼±À» ±×¸°´Ù.

Description

int imageline(int im, int x1, int y1, int x2, int y2, int col);

ImageLine draws a line from x1,y1 to x2,y2 (top left is 0,0) in image im of color col.

See also **imagecreate()** and **imagecolorallocate()**.

ImageLoadFont

ImageLoadFont - - »õ Æù¾®¸¦ loadÇÑ´Ù.

Description

int imageloadfont(string file);

ImageLoadFont loads a user-defined bitmap font and returns an identifier for the font (that is always greater than 5, so it will not conflict with the built-in fonts).

The font file format is currently binary and architecture dependent. This means you should generate the font files on the same type of CPU as the machine you are running PHP on.

Table 1. Font file format

byte position	C data type	description
byte 0-3	int	number of characters in the font
byte 4-7	int	value of first character in the font (often 32 for space)
byte 8-11	int	pixel width of each character
byte 12-15	int	pixel height of each character
byte 16-	char	array with character data, one byte per pixel in each character, for a total of (nchars*width*height) bytes.

See also **ImageFontWidth()** and **ImageFontHeight()**.

ImagePolygon

ImagePolygon - - ´Ù°¢ÇüÀ» ±×¸°´Ù.

Description

int imagepolygon(int im, int points, int num_points, int col);

ImagePolygon creates a polygon in image id. points is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. num_points is the total number of vertices.

See also **imagecreate()**.

ImageRectangle

ImageRectangle - - Á÷»ç°¢ÇüÀ» ±×¸°´Ù.

Description

int imagerectangle(int im, int x1, int y1, int x2, int y2, int col);

ImageRectangle creates a rectangle of color col in image im starting at upper left coordinate x1,y1 and ending at bottom right coordinate x2,y2. 0,0 is the top left corner of the image.

ImageSetPixel

ImageSetPixel - - ÇÑ¼ÀÇ »öÀ» ¼³Á¤ÇÑ´Ù.

Description

int imagesetpixel(int im, int x, int y, int col);

ImageSetPixel draws a pixel at x,y (top left is 0,0) in image im of color col.

See also **imagecreate()** and **imagecolorallocate()**.

ImageString

ImageString - - ¹®ÀÚ¿À» ¼öÆòÀ¸·Î ±×¸°´Ù.

Description

int imagestring(int im, int font, int x, int y, string s, int col);

ImageString draws the string s in the image identified by im at coordinates x,y (top left is 0,0) in color col. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

ImageStringUp

ImageStringUp - - ¹®ÀÚ¿À» ¼öÁ÷À¸·Î ±×¸°´Ù.

Description

int imagestringup(int im, int font, int x, int y, string s, int col);

ImageStringUp draws the string s vertically in the image identified by im at coordinates x,y (top left is 0,0) in color col. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

ImageSX

ImageSX - - ±×¸²ÀÇ ÆøÀ» ±¸Ñ´Ù.

Description

int imagesx(int im);

ImageSX returns the width of the image identified by im.

See also **imagecreate()** and **imagesy()**.

ImageSY

ImageSY - - ±×¸²ÀÇ ³ôÀÌ¸¦ ±¸ÇÑ´Ù.

Description

int imagesy(int im);

ImageSY returns the height of the image identified by im.

See also **imagecreate()** and **imagesx()**.

ImageTTFBBox

ImageTTFBBox - - TrueType font¸¦ »ç¿ëÇÏ¿© ¹®ÀÚ¿ ¼²°í, ÁÖÀ§ÀÇ ¹®ÀÚ ÁÖÀ§¿¡ °æ°è¼±À» ±×¸°´Ù.

Description

array ImageTTFBBox(int size, int angle, string fontfile, string text);

This function calculates and returns the bounding box in pixels a TrueType text.

text The string to be measured.

size The font size.

fontfile The name of the TrueType font file. (Can also be an URL.)

angle Angle in degrees in which *text* will be measured.

ImageTTFBBox() returns an array with 8 elements representing four points making the bounding box of the text:

0	lower left corner, X position
1	lower left corner, Y position
2	lower right corner, X position
3	lower right corner, Y position
4	upper right corner, X position
5	upper right corner, Y position
6	upper left corner, X position
7	upper left corner, Y position

The points are relative to the *text* regardless of the angle, so "upper left" means in the top left-hand corner seeing the text horizontallty.

This function requires both the GD library and the Freetype library.

See also **ImageTTFText()**.

ImageTTFText

ImageTTFText - - TrueType ÆùÆ®¸¦ »ç¿ëÇÏ¿© ¹®ÀÚ¿À» ¾´´Ù.

Description

array ImageTTFText(int im, int size, int angle, int x, int y, int col, string fontfile, string text);

ImageTTFText draws the string *text* in the image identified by *im*, starting at coordinates *x,y* (top left is 0,0), at an angle of *angle* in color *col*, using the TrueType font file identified by *fontfile*.

The coordinates given by *x,y* will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the **ImageString()**, where x,y define the upper-right corner of the first character.

angle is in degrees, with 0 degrees being left-to-right reading text (3 o'clock direction), and higher values representing a counter-clockwise rotation. (i.e., a value of 90 would result in bottom-to-top reading text).

fontfile is the path to the TrueType font you wish to use.

text is the text string which may include UTF-8 character sequences (of the form: {) to access characters in a font beyond the first 255.

col is the color index. Using the negative of a color index has the effect of turning off antialiasing.

ImageTTFText() returns an array with 8 elements representing four points making the bounding box of the text. The order of the points is upper left, upper right, lower right, lower left. The points are relative to the text regardless of the angle, so "upper left" means in the top left-hand corner when you see the text horizontallty.

This example script will produce a black GIF 400x30 pixels, with the words "Testing..." in white in the font Arial.
Example 1. ImageTTFText

```
<?php
Header("Content-type: image/gif");
$im = imagecreate(400,30);
$black = ImageColorAllocate($im, 0,0,0);
$white = ImageColorAllocate($im, 255,255,255);
ImageTTFText($im, 20, 0, 10, 20, $white, "/path/arial.ttf", "Testing... Omega: &#937;");
ImageGif($im);
ImageDestroy($im);
?>
```

This function requires both the GD library and the FreeType< library.

See also **ImageTTFBBox()**.

ImageColorAt

ImageColorAt - - ÇØ´ç pixel »ö±òÀÇ index¸¦ ±¸Ñ´Ù.

Description

int imagecolorat(int im, int x, int y);

Returns the index of the color of the pixel at the specified location in the image.

See also **imagecolorset()** and **imagecolorsforindex()**.

ImageColorClosest

ImageColorClosest - - ÁöÁ¤µÈ »ö°ú °¡Àå °¡±î¿î »öÀÇ index¸¦ ±¸Ñ´Ù.

Description

int imagecolorclosest(int im, int red, int green, int blue);

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value.

The "distance" between the desired color and each color in the palette is calculated as if the RGB values represented points in three-dimensional space.

See also **imagecolorexact()**.

ImageColorExact

ImageColorExact - - ÁöÁ¤µÈ »öÀÇ index¸¦ ±¸Ñ´Ù.

Description

int imagecolorexact(int im, int red, int green, int blue);

Returns the index of the specified color in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also **imagecolorclosest()**.

ImageColorResolve

ImageColorResolve - - Æ¯Á¤»öÀÌ³ª ±×¿¡ °¡Àå °¡±î¿î »öÀÇ index¸¦ ±¸Ñ´Ù.

Description

int imagecolorresolve(int im, int red, int green, int blue);

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also **imagecolorclosest()**.

ImageColorSet

ImageColorSet - - ÁöÁ¤ÇÑ »öÀ» Æ¯Á¤ palette index·Î Á¤ÇÑ´Ù.

Description

bool imagecolorset(int im, int index, int red, int green, int blue);

This sets the specified index in the palette to the specified color. This is useful for creating flood - fill - like effects in paletted images without the overhead of performing the actual flood - fill.

See also **imagecolorat()**.

ImageColorsForIndex

ImageColorsForIndex - - ÁöÁ¤ÇÑ indexÀÇ »öÀ» ±¸ÇÑ´Ù.

Description

array imagecolorsforindex(int im, int index);

This returns an associative array with red, green, and blue keys that contain the appropriate values for the specified color index.

See also **imagecolorat()** and **imagecolorexact()**.

ImageColorsTotal

ImageColorsTotal - - ±×¸²ÀÇ palette¿¡ ÀÖ´Â »öÀÇ °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int imagecolorstotal(int im);

This returns the number of colors in the specified image's palette.

See also **imagecolorat()** and **imagecolorsforindex()**.

XIX. IMAP Functions

Table of Contents

ÀÌ ÇÔ¼öµéÀ» µ¿ÀÛ½ÃÅ°·Á¸é PHP¸¦ - - with - imap ¿É¼ÇÀ» ÁÖ°í ÄÄÆÄÀÏ ÇÏ¿©¾ß ÇÑ´Ù. ¶ÇÇÑ ÀÌ°ÍÀÌ ¼³Ä¡µÇ±â À§ÇØ¼´Â c - client library°¡ ÇÊ¿äÇÏ´Ù. °¡Àå ÃÖ½Å ¹öÀüÀº ftp://ftp.cac.washington.edu/imap/¿¡ ÀÖÀ¸¹Ç·Î, ±¸ÇÏ¿© ÄÄÆÄÀÏÇÏ¸é µÈ´Ù. ÄÄÆÄÀÏ ÈÄ c - client/c - client.a¸¦ /usr/local/lib³ª ¿©·¯°ÐÀÌ ¼³Á¤ÇØ ³õÀº link pathÀÇ µð·ºÅä¸®·Î °¹»çÇÏ°í, c - client/rfc822.h¿Í mail.h, linkage.h¸¦ /usr/local/include³ª ¿©·¯°ÐÀÇ include path¿¡ °¹»çÇØ µÐ´Ù.

imap_append

imap_append - - ÁöÁ¤ÇÑ ÆíÁöÇÔ¿¡ ¹®ÀÚ¿ ¸Þ¼¼Áö¸¦ µ¡ºÙÀÎ´Ù.

Description

int imap_append(int imap_stream, string mbox, string message, stringflags);

Returns true on sucess, false on error.

imap_append() appends a string message to the specified mailbox *mbox*. If the optional *flags* is specified, writes the *flags* to that mailbox also.

When talking to the Cyrus IMAP server, you must use " \ r \ n" as your end - of - line terminator instead of " \ n" or the operation will fail.

imap_base64

imap_base64 - - BASE64·Î encodeµÈ text¸¦ decodeÇÑ´Ù.

Description

string imap_base64(string text);

imap_base64() function decodes BASE - 64 encoded text. The decoded message is returned as a string.

imap_body

imap_body - - message body¸¦ ÀÐ´Â´Ù.

Description

string imap_body(int imap_stream, int msg_number, int flags);

imap_body() returns the body of the message, numbered *msg_number* in the current mailbox. The optional *flags* are a bit mask with one or more of the following:

FT_UID - The msgno is a UID

FT_PEEK - Do not set the \ Seen flag if not already set

FT_INTERNAL - The return string is in internal format, will not canonicalize to CRLF.

imap_check

imap_check - - Çö mailbox¸¦ °Ë»çÇÑ´Ù.

Description

array imap_check(int imap_stream);

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_check()** function checks the current mailbox status on the server and returns the information in an object with following properties.

Date : date of the message
Driver : driver
Mailbox : name of the mailbox
Nmsgs : number of messages
Recent : number of recent messages

imap_close

imap_close - - IMAP streamÀ» ´Ý´Â´Ù.

Description

int imap_close(int imap_stream, int flags);

Close the imap stream. Takes an optional *flag* CL_EXPUNGE, which will silently expunge the mailbox before closing.

imap_createmailbox

imap_createmailbox - - »õ ÆíÁöÇÔÀ» ¸¸µç´Ù.

Description

int imap_createmailbox(int imap_stream, string mbox);

imap_createmailbox() creates a new mailbox specified by *mbox*.

Returns true on success and false on error.

imap_delete

imap_delete - - ÇöÀç ÆíÁöÇÔ¿¡¼ ¾î¶² messge¿¡ »èÁ¦ ¸¶Å©¸¦ ÇÑ´Ù.

Description

int imap_delete(int imap_stream, int msg_number);

Returns true.

imap_delete() function marks message pointed by *msg_number* for deletion. Actual deletion of the messages is done by **imap_expunge()**.

imap_deletemailbox

imap_deletemailbox - - ÆíÁöÇÔÀ» Áö¿î´Ù.

Description

int imap_deletemailbox(int imap_stream, string mbox);

imap_deletemailbox() deletes the specified mailbox.

Returns true on success and false on error.

imap_expunge

imap_expunge - - »èÁ¦ ¸¶Å©°¡ µÇ¾î ÀÖ´Â ¸ðµç ¸Þ¼¼Áö¸¦ Áö¿î´Ù.

Description

int imap_expunge(int imap_stream);

imap_expunge() deletes all the messages marked for deletion by **imap_delete()**.

Returns true.

imap_fetchbody

imap_fetchbody - - messageÀÇ bodyÁß particular sectionÀ» °¡Áö°í ¿Â´Ù.

Description

string imap_fetchbody(int imap_stream, int msg_number, int part_number, flags flags);

This function causes a fetch of a particular section of the body of the specified messages as a text string and returns that text string. The section specification is a string of integers delimited by period which index into a body part list as per the IMAP4 specification. Body parts are not decoded by this function.

The options for **imap_fetchbody ()**e a bitmask with one or more of the following

FT_UID - The msgono is a UID

FT_PEEK - Do not set the \ Seen flag if not already set

FT_UID - The return string is in "internal" format, without any attempt to canonicalize CRLF

imap_fetchstructure

imap_fetchstructure - - particular messageÀÇ ±¸Á¶¸¦ ÀÐ¾î¿Â´Ù.

Description

array imap_fetchstructure(int imap_stream, int msg_number);

This function causes a fetch of all the structured information for the given msg_number. The returned value is an object with following elements.

type, encoding, ifsubtype, subtype, ifdescription, description, ifid,
id, lines, bytes, ifparameters

It also returns an array of objects called parameters[]. This object has following properties.

attribute, value

In case of multipart, it also returns an array of objects of all the properties, called parts[].

imap_header

imap_header - - messageÀÇ header¸¦ ÀÐ¾î¿Â´Ù.

Description

object imap_header(int imap_stream, int msg_number, int fromlength, int subjectlength, int defaulthost);

This function returns an object of various header elements

remail,date,Date,subject,Subject,in_reply_to,message_id,newsgroups,
followup_to,references
toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing:)
  personal
  adl
  mailbox
  host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing:)
  personal
  adl
  mailbox
  host

ccaddress (full cc: line, up to 1024 characters)
cc[] (returns an array of objects from the Cc line, containing:)
  personal
  adl
  mailbox
  host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing:)
  personal
  adl
  mailbox
  host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line, containing:)
  personal
  adl
  mailbox
  host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing:)
  personal
  adl
  mailbox
  host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line, containing:)
  personal
  adl
  mailbox
  host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *fromlength* characters)
fetchsubject (subject line formatted to fit *subjectlength* characters)

imap_headers

imap_headers - - ÇÑ ÆíÁöÇÔÀÇ ¸ðµç messageµéÀÇ header¸¦ ÀÐ¾î¿Â´Ù.

Description

array imap_headers(int imap_stream);

Returns an array of string formatted with header info. One element per mail message.

imap_listmailbox

imap_listmailbox - - ÆíÁöÇÔÀÇ ¸ñ·ÏÀ» ÀÐ¾î ¿Â´Ù.

Description

array imap_listmailbox(int imap_stream, string ref, string pat);

Returns an array containing the names of the mailboxes.

imap_listsubscribed

imap_listsubscribed - - ¸ðµç subscribed ÆíÁöÇÔ¼ö¸ñ·ÏÀ» ÀÐ¾î ¿Â´Ù.

Description

array imap_listsubscribed(int imap_stream, string ref, string pattern);

Returns an array of all the mailboxes that you have subscribed. The *ref* and *pattern* arguments specify the base location to search from and the pattern the mailbox name must match.

imap_mail_copy

imap_mail_copy - - Æ¯Á¤ messageµéÀ» ´Ù¸¥ ÆíÁöÇÔ¸·Î º¹»çÇÑ´Ù.

Description

int imap_mail_copy(int imap_stream, string msglist, string mbox, int flags);

Returns true on success and false on error.

Copies mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers.

flags is a bitmask of one or more of

CP_UID - the sequence numbers contain UIDS

CP_MOVE - Delete the messages from the current mailbox after copying

imap_mail_move

imap_mail_move - - Æ¯Á¤ messageµéÀ» ´Ù¸¥ ÆíÁöÇÔ¸·Î ¿Å±ä´Ù.

Description

int imap_mail_move(int imap_stream, string msglist, string mbox);

Moves mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers.

Returns true on success and false on error.

imap_num_msg

imap_num_msg - - ÇöÀç ÆíÁöÇÔ¼ö Þ¼¼Áö °³¼ö¸¦ ±¸Ñ´Ù.

Description

int imap_num_msg(void);

Return the number of messages in the current mailbox.

imap_num_recent

imap_num_recent - - ÇöÀç ÆíÁöÇÔ¿Ù ÃÖ±Ù ¸Þ¼¼Áö °³¼ö¸¦ ±¸Ñ´Ù.

Description

int imap_num_recent(int imap_stream);

Returns the number of recent messages in the current mailbox.

imap_open

imap_open - - ÇÑ ÆíÁöÇÔ¿¡ ´ëÇØ IMAP streamÀ» ¿¬´Ù.

Description

int imap_open(string mailbox, string username, string password, int flags);

Returns an IMAP stream on success and false on error. This function can also be used to open streams to POP3 and NNTP servers. To connect to an IMAP server running on port 143 on the local machine, do the following:

```
$mbox = imap_open("{localhost:143}INBOX","user_id","password");
```

To connect to a POP3 server on port 110 on the local server, use:

```
$mbox = imap_open("{localhost/pop3:110}INBOX","user_id","password");
```

To connect to an NNTP server on port 119 on the local server, use:

```
$nntp = imap_open("{localhost/nntp:119}comp.test","","");
```

To connect to a remote server replace "localhost" with the name or the IP address of the server you want to connect to.

The options are a bit mask with noe or more of the following:

OP_READONLY - Open mailbox read - only

OP_ANONYMOUS - Dont use or update a .newsrc for news

OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox

CL_EXPUNGE - Expunge mailbox automatically upon mailbox close

imap_ping

imap_ping - - IMAP streamÀÌ ¿©ÀüÈ÷ activeÀÎÁö °Ë»çÇÑ´Ù.

Description

int imap_ping(int imap_stream);

Returns true if the stream is still alive, false otherwise.

imap_ping() function pings the stream to see it is still active. It may discover new mail; this is the preferred method for a periodic "new mail check" as well as a "keep alive" for servers which have inactivity timeout.

imap_renamemailbox

imap_renamemailbox - - ÆíÁöÇÔÀÇ ÀÌ¸§À» ¹Ù²Û´Ù.

Description

int imap_renamemailbox(int imap_stream, string old_mbox, string new_mbox);

This function renames on old mailbox to new mailbox.

Returns true on success and false on error.

imap_reopen

imap_reopen - - ±âÁ¸¿¡ ¿-·ÁÀÖ´Â IMAP streamÀ» »õ ÆíÁöÇÔÀ¸·Î ¿¬´Ù.

Description

int imap_reopen(string imap_stream, string mailbox, string [flags]);

Returns true on success and false on error.

This function reopens the specified stream to new mailbox.

the options are a bit mask with noe or more of the following:

OP_READONLY - Open mailbox read - only

OP_ANONYMOUS - Dont use or update a .newsrc for news

OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox

CL_EXPUNGE - Expunge mailbox automatically upon mailbox close

imap_subscribe

imap_subscribe - - »õ ÆíÁöÇÔÀ» subscribeÇÑ´Ù.

Description

`int imap_subscribe`(int imap_stream, string mbox);

Subscribe to a new mailbox.

Returns true on success and false on error.

imap_undelete

imap_undelete - - »èÁ¦ Ç¥½ÃµÈ messageÀÇ »èÁ¦ Ç¥½Ã¸¦ Áö¿î´Ù .

Description

`int imap_undelete`(int imap_stream, int msg_number);

This function removes the deletion flag for a specified message, which is set by **imap_delete()**.

Returns true on success and false on error.

imap_unsubscribe

imap_unsubscribe - - ÆíÁöÇÔÀ» unsubscribeÇÑ´Ù.

Description

`int imap_unsubscribe`(int imap_stream, string mbox);

Unsubscribe from a specified mailbox.

Returns true on success and false on error.

imap_qprint

imap_qprint - - quoted - printable ¹®ÀÚ¿À» 8 bit ¹®ÀÚ¿·Î ¹Ù²Û´Ù.

Description

`string imap_qprint`(string string);

Convert a quoted - printable string to an 8 bit string

Returns an 8 bit (binary) string

imap_8bit

imap_8bit - - 8 bit ¹®ÀÚ¿À» quoted - printable ¹®ÀÚ¿·Î ¹Ù²Û´Ù.

Description

`string imap_8bit`(string string);

Convert an 8bit string to a quoted - printable string.

Returns a quoted - printable string

imap_binary

imap_binary - - 8bit ¹®ÀÚ¿-À» base64 ¹®ÀÚ¿-·Î ¹Ù²Û´Ù.

Description

`string imap_binary`(string string);

Convert an 8bit string to a base64 string.

Returns a base64 string

imap_scanmailbox

imap_scanmailbox - - mailboxµéÀÇ list¸¦ ÀÐ°í, °Ë»öÇÒ ¹®ÀÚ¿-À» mailboxÀÇ text¿¡¼- ÃëÇÑ´Ù.

takes a string to search for in the text of the mailbox

Description

`array imap_scanmailbox`(int imap_stream, string string);

Returns an array containing the names of the mailboxes that have *`string`* in the text of the mailbox.

imap_mailboxmsginfo

imap_mailboxmsginfo - - ÇöÀç mailbox¿¡ ´ëÇÑ Á¤º¸¸¦ ±¸Ñ´Ù.

Description

`array imap_mailboxmsginfo`(int imap_stream);

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_mailboxmsginfo()** function checks the current mailbox status on the server and returns the information in an object with following properties.

Date : date of the message
Driver : driver
Mailbox : name of the mailbox
Nmsgs : number of messages
Recent : number of recent messages
Unread : number of unread messages
Size : mailbox size

imap_rfc822_write_address

imap_rfc822_write_address - - ÁÖ¾îÁø mailbox¿Í host, personal info·Î ÀûÇÕÇÑ ¸ð¾çÀÇ email address¸¦ ¸¸µé¾î ³½´Ù.

Returns a properly formatted email address given the mailbox, host, and personal info.

Description

`string imap_rfc822_write_address`(string mailbox, string host, string personal);

Returns a properly formatted email address given the mailbox, host, and personal info.

imap_rfc822_parse_adrlist

imap_rfc822_parse_adrlist - - address ¹®ÀÚ¿-À» parsingÇÑ´Ù.

Description

string imap_rfc822_parse_adrlist(string address, string default_host);

This function parses the address tring and for each address, returns an array of objects. The 4 objects are:

mailbox - the mailbox name (username)
host - the host name
personal - the personal name
adl - at domain source route

imap_setflag_full

imap_setflag_full - - messageµéÀÇ ¿©·¯ flag¸¦ ¼³Á¤(set)ÇÑ´Ù.

Description

string imap_setflag_full(int stream, string sequence, string flag, string options);

This function causes a store to add the specified flag to the flags set for the messages in the specified sequence.

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

imap_clearflag_full

imap_clearflag_full - - messageµéÀÇ ¿©·¯ flag¸¦ clear ÇÑ´Ù.

Description

string imap_clearflag_full(int stream, string sequence, string flag, string options);

This function causes a store to delete the specified flag to the flags set for the messages in the specified sequence.

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

imap_sort

imap_sort - - ¸Þ½ÃÁöµéÀ» ÁÖ¼îÁø ¹æ½Ä´ë·Î Á¤·ÄÇÑ ¸Þ½ÃÁö ¹øÈ£ÀÇ ¹è¿-À» ±¸Ç´Ù.

Description

string imap_sort(int stream, int criteria, int reverse, int options);

Returns an array of message numbers sorted by the given parameters

Rev is 1 for reverse - sorting.

Criteria can be one (and only one) of the following:

SORTDATE message Date
SORTARRIVAL arrival date
SORTFROM mailbox in first From address
SORTSUBJECT message Subject
SORTTO mailbox in first To address
SORTCC mailbox in first cc address
SORTSIZE size of message in octets

The flags are a bitmask of one or more of the following:

SE_UID Return UIDs instead of sequence numbers
SE_NOPREFETCH Don't prefetch searched messages.

imap_fetchheader

imap_fetchheader - - messageÀÇ Çì´õ¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

stringimap_fetchheader(int imap_stream, int msgno, int flags);

This function causes a fetch of the complete, unfiltered RFC 822 format header of the specified message as a text string and returns that text string.

The options are:

FT_UID The msgno argument is a UID
FT_INTERNAL The return string is in "internal" format, without any attempt to canonicalize to CRLF newlines
FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the same time. This avoids an extra RTT on an IMAP connection if a full message text is desired (e.g. in a "save to local file" operation)

imap_uid

imap_uid - - ¼±ÅÃÇÑ messageÀÇ ÀÏ·Ã ¹øÈ£(sequence number)ÀÎ UID¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

string imap_uid(string mailbox, int msgno);

This function returns the UID for the given message sequence number

---

XX. PHP options & information

Table of Contents

error_log

error_log - - ¿¡·¯ ¸Þ¼¼Áö¸¦ ÁöÁ¤ÇÑ °÷¿¡ º¸³½´Ù.

Description

int error_log(string message, int message_type, string [destination], string [extra_headers]);

Sends an error message to the web server's error log, a TCP port or to a file. The first parameter, *message*, is the error message that should be logged. The second parameter, *message_type* says where the message should go:

Table 1. error_log() log types

0	*message* is sent to PHP's system logger, using the Operating System's system logging mechanism or a file, depending on what the error_log configuration directive is set to.
1	*message* is sent by email to the address in the *destination* parameter. This is the only message type where the fourth parameter, *extra_headers* is used. This message type uses the same internal function as **Mail()** does.
2	*message* is sent through the PHP debugging connection. This option is only available if remote debugging has been enabled. In this case, the *destination* parameter specifies the host name or IP address and optionally, port number, of the socket receiving the debug information.
3	*message* is appended to the file *destination*.

Example 1. error_log() examples

```
// Send notification through the server log if we can not
// connect to the database.
if (!Ora_Logon($username, $password)) {
    error_log("Oracle database not available!", 0);
}
// Notify administrator by email if we run out of FOO
if (!($foo = allocate_new_foo()) {
    error_log("Big trouble, we're all out of FOOs!", 1,
              "operator@mydomain.com");
}
// other ways of calling error_log():
error_log("You messed up!", 2, "127.0.0.1:7000");
error_log("You messed up!", 2, "loghost");
error_log("You messed up!", 3, "/var/tmp/my-errors.log");
```

error_reporting

error_reporting - - report µÉ PHP ¿¡·¯µéÀ» Á¤ÇÑ´Ù.

Description

```
int error_reporting(int [level]);
```

Sets PHP's error reporting level and returns the old level. The error reporting level is a bitmask of the following values (follow the links for the internal values to get their meanings):

Table 1. error_reporting() bit values

value	internal name
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING

getenv

getenv - - È¯°æ º¯¼ö(environment variable)ÀÇ °ªÀ» ±¸Ñ´Ù.

Description

```
string getenv(string varname);
```

Returns the value of the environment variable *varname*, or false on an error.

```
$ip = getenv("REMOTE_ADDR"); // get the ip number from the user
```

get_cfg_var

get_cfg_var - - PHP ¼³Á¤ ¿É¼Ç(configuration option)ÀÇ °ªÀ» ±¸Ñ´Ù.

Description

```
string get_cfg_var(string varname);
```

Returns the current value of the PHP configuration variable specified by *varname*, or false if an error occurs.

It will not return configuration information set when the PHP was compiled, or read from an Apache configuration file (using the php3_configuration_option directives).

To check whether the system is using a php3.ini file, try retrieving the value of the cfg_file_path configuration setting. If this is available, a php3.ini file is being used.

get_current_user

get_current_user - - Çö PHP scriptÀÇ owner ÀÌ¸§À» ±¸ÇÑ´Ù.

Description

string get_current_user(void);

Returns the name of the owner of the current PHP script.

See also **getmyuid()**, **getmypid()**, **getmyinode()**, and **getlastmod()**.

get_magic_quotes_gpc

get_magic_quotes_gpc - - magic quotes gpcÀÇ ÇöÀç ¼³Á¤ »óÅÂ¸¦ ¾ò´Â´Ù.

Description

long get_magic_quotes_gpc(void);

Returns the current active configuration setting of magic_quotes_gpc. (0 for off, 1 for on)

See also **get_magic_quotes_runtime()**, **set_magic_quotes_runtime()**.

get_magic_quotes_runtime

get_magic_quotes_runtime - - magic_quotes_runtimeÀÇ ÇöÀç ¼³Á¤ »óÅÂ¸¦ ¾ò´Â´Ù.

Description

long get_magic_quotes_runtime(void);

Returns the current active configuration setting of magic_quotes_runtime. (0 for off, 1 for on)

See also **get_magic_quotes_gpc()**, **set_magic_quotes_runtime()**.

getlastmod

getlastmod - - Çö ¹®¼¸¦ ¸¶Áö¸·À¸·Î ¼öÁ¤ÇÑ ½Ã°£À» ±¸ÇÑ´Ù.

Description

int getlastmod(void);

Returns the time of the last modification of the current page. The value returned is a Unix timestamp, suitable for feeding to **date()**. Returns false on error.
Example 1. getlastmod() example

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: ".date( "F d Y H:i:s.", getlastmod() );
```

See alse **date()**, **getmyuid()**, **get_current_user()**, **getmyinode()**, and **getmypid()**.

getmyinode

getmyinode - - Çö scriptÀÇ inode¸¦ ±¸ÇÑ´Ù.

Description

int getmyinode(void);

Returns the current script's inode, or false on error.

See also **getmyuid()**, **get_current_user()**, **getmypid()**, and **getlastmod()**.

getmypid

getmypid -- PHPÀÇ process ID¸¦ ±¸ÇÑ´Ù.

Description

int getmypid(void);

Returns the current PHP process ID, or false on error.

Note that when running as a server module, separate invocations of the script are not guaranteed to have distinct pids.

See also **getmyuid()**, **get_current_user()**, **getmyinode()**, and **getlastmod()**.

getmyuid

getmyuid -- PHP scriptÀÇ ownerÀÇ UID °ªÀ» ±¸ÇÑ´Ù..

Description

int getmyuid(void);

Returns the user ID of the current script, or false on error.

See also **getmypid()**, **get_current_user()**, **getmyinode()**, and **getlastmod()**.

getrusage

getrusage -- ÇöÀç ÀÚ¿øÀÇ »ç¿ë ÇöÈ²À» ¾ò´Â´Ù.

Description

array getrusage(int [who]);

This is an interface to getrusage(2). It returns an associative array containing the data returned from the system call. If who is 1, getrusage will be called with RUSAGE_CHILDREN. All entries are accessible by using their documented field names.

Example 1. Getrusage Example

```
$dat = getrusage();
echo $dat["ru_nswap"];         # number of swaps
echo $dat["ru_majflt"];        # number of page faults
echo $dat["ru_utime.tv_sec"];  # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```

See your system's man page for more details.

phpinfo

phpinfo -- ¿©·¯°¡ÁöÀÇ PHP informationÀ» ÃâÂÇÑ´Ù.

Description

int phpinfo(void);

Outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the GNU Public License.

See also **phpversion()**.

phpversion

phpversion - - Çö PHP versionÀ» ±¸ÇÑ´Ù.

Description

string phpversion(void);

Returns a string containing the version of the currently running PHP parser.
Example 1. phpversion() example

```
// prints e.g. 'Current PHP version: 3.0rel-dev'
echo "Current PHP version: ".phpversion();
```

See also **phpinfo()**.

putenv

putenv - - È¯°æ º¯¼ö(environment variable)ÀÇ °ªÀ» ¼³Á¤ÇÑ´Ù.

Description

void putenv(string setting);

Adds *setting* to the environment.
Example 1. Setting an Environment Variable

```
putenv("UNIQID=$uniqid");
```

set_magic_quotes_runtime

set_magic_quotes_runtime - - Set the current active configuration setting of magic_quotes_runtime.

Description

long set_magic_quotes_runtime(int new_setting);

Set the current active configuration setting of magic_quotes_runtime. (0 for off, 1 for on)

See also **get_magic_quotes_gpc()**, **get_magic_quotes_runtime()**.

set_time_limit

set_time_limit - - ½ÇÇà¿¡ °É¸®´Â ½Ã°£À» ÇÑ¾ÅÅ²´Ù. (limit the maximum execution time)

Description

void set_time_limit(int seconds);

Set the number of seconds a script is allowed to run. If this is reached, the script returns a fatal error. The default limit is 30 seconds or, if it exists, the max_execution_time value defined in php3.ini. If seconds is set to zero, no time limit is imposed.

When called, **set_time_limit()** restarts the timeout counter from zero. In other words, if the timeout is the default 30 seconds, and 25 seconds into script execution a call such as set_time_limit(20) is made, the script will run for a total of 45 seconds before timing out.

XXI. Informix Functions

Table of Contents

ifx_getsqlca
ifx_fetch_row
ifx_htmltbl_result
ifx_fieldtypes
ifx_fieldproperties
ifx_num_fields
ifx_num_rows
ifx_free_result
ifx_create_char
ifx_free_char
ifx_update_char
ifx_get_char
ifx_create_blob
ifx_copy_blob
ifx_free_blob
ifx_get_blob
ifx_update_blob
ifx_blobinfile_mode
ifx_textasvarchar
ifx_byteasvarchar
ifx_nullformat
ifxus_create_slob
ifx_free_slob
ifxus_close_slob
ifxus_open_slob
ifxus_tell_slob
ifxus_seek_slob
ifxus_read_slob
ifxus_write_slob

InformixÀÇ Online (ODS) 7.x¿Í SE 7.x, Universal Server (IUS) 9.x¸¦ À§ÇÑ µå¶óÀÌ¹ö´Â "functions/ifx.ec"¿Í "functions/php3_ifx.h"¿¡ ±¸ÇöµÇ¾î ÀÖ´Ù. ÀÌ±ÛÀÌ ¾²¿©Áú ´ç½Ã ¿ÏÀüÇÑ BLOB Áö¿øÀ» Æ÷ÇÔÇÏ¿©, ODS 7.2¿¡ ´ëÇÑ Áö¿ø´Â °ÅÀÇ ¿Ï¼ºµÇ¾ú´Ù. IUS 9.1¿¡ ´ëÇÑ Áö¿øÀº ºÎºĐÀûÀ¸·Î¸¸ ¿Ï·áµÇ¾ú´Ù. »õ·Î¿î µ¥ÀÌÅÍ Å¸ÀÔµéÀº ¿Ï·áµÇ¾úÀ¸³ª, SLOBS¿¡ ´ëÇÑ Áö¿øÀº ¾ÆÁ÷ ÁøÇàÁßÀÌ´Ù.

¼³Á¤½Ã ÁÖÀÇ »çÇ× (Configuration notes) :

"configure" ½ºÅ©¸³Æ®¸¦ ½ÇÇàÇÏ±â Àü¿¡, ¹Ýµå½Â "INFORMIXDIR" È¯°æ º¯¼ö¸¦ ¼³Á¤ÇØ ³õ¾Æ¾ß ÇÑ´Ù.

ÀÌ·¸°Ô ÇØ ³õ°í "configure --with_informix=yes"·Î ½ºÅ©¸³Æ® ÆÄÀÏÀ» ½ÇÇàÇÏ¸é, configure ½ºÅ©¸³Æ®´Â ¶óÀÌºê·¯¸®¿Í include°¡ ÀÖ´Â µð·ºÅä¸®¸¦ ÀÚµ¿À¸·Î ã£´Â´Ù. ¸¸¾à ¿©·¯ºĐÀÌ ¼ÕÀ¸·Î ¼³Á¤ ³»¿ëÀ» ÁöÁ¤ÇÏ°í ½Í´Ù¸é "IFX_LIBDIR", "IFX_LIBS", "IFX_INCDIR"ÀÇ È¯°æº¯¼ö¿¡ ¿øÇÏ´Â °ªÀ» ÁöÁ¤ÇÏ¸é µÈ´Ù. ¶ÇÇÑ, configure ½ºÅ©¸³Æ®´Â ¿©·¯ºĐÀÌ »ç¿ëÁßÀÎ InformixÀÇ ¹öÀüÀ» Ã¼Å©ÇÑ´Ù. ¸¸¾à ¿©·¯ºĐÀÌ Informix ¹öÀüÀÌ 9.00ÀÌ»óÀÌ¶ó¸é ÀÌ °ªÀÌ "HAVE_IFX_IUS" ¶ó´Â conditional compilation variable¿¡ ¼³Á¤ÇÑ´Ù.

BLOB »ç¿ë¼Ã ÁÖÀÇ »çÇ× (Some notes on the use of BLOBs) :

ÇöÀç ¹öÀü(September 18, 1998)Àº select/insert/update¿¡¼- BLOB Ä®·³À» ¿Ï°®ÇÏ°Ô Áö¿øÇÑ´Ù.

BLOB´Â °¸Åë Á¤¼ö°ªÀÎ BLOB identifier¸¦ »ç¿ëÇÏ¿© ÁÖ¼ÒÈ-(addressed)µÈ´Ù. Select ÁúÀÇ´Â ¸ðµç BYTE¿Í TEXT Ä®·³¿¡ ´ëÇÑ "blob id"¸¦ ¹ÝÈ¯ÇÑ´Ù. ¿©·¯ºĐÀÌ "ifx_blobinfile(0);"¸¦ »ç¿ëÇÏ¿© BLOB¸¦ ¸Þ¸ð¸®¿¡¼- °¡Á®¿À±â·Î °áÁ¤ÇØ µÎ¼ú´Ù¸é, ¿©·¯ºĐÀº "string_var = ifx_get_blob($blob_id);" °°Àº ¸í·ÉÀ» »ç¿ëÇÏ¿© ±× ³»¿ëÀ» ¾ò¾î¿Ã ¼ö ÀÖ´Ù. ¸¸¾à, "ifx_blobinfile(1);"¸¦ »ç¿ëÇÏ¿© ÆÄÀÏ¿¡ ÀÖ´Â BLOB Ä®·³ÀÇ ³»¿ëÀ» ÀÚÁ®¿À±â·Î ÇÏ¿´Ù¸é, "ifx_get_blob($blob_id);"¸¦ »ç¿ëÇÏ¿© ÇØ´ç ÆÄÀÏ ÀÌ¸§À» ¾òÀ» ¼ö ÀÖ´Ù. ÀÌ¶§ ¾òÀº ÆÄÀÏ ÀÌ¸§Àº ÀÏ¹ÝÛÀÎ ÆÄÀÏ I/O ¹æ½ÄÀ» »ç¿ëÇÏ¿© ±× ³»¿ëÀ» ÀĐ¾î¿Ã ¼ö ÀÖ´Ù.

insert/update ÁúÀÇÀÇ °æ¿ì ¿©·¯ºĐÀº "ifx_create_blob(..);"¸¦ »ç¿ëÇÏ¿© "blob id(µé)"¸¦ ¼Õ¼ö ¸¸¾î ÁÖ¾ß ÇÑ´Ù. ¸¸µç blob idµéÀº ¹è¿-¿¡ ÁúÀåÇÑ ËÄ¿¡, blob Ä®·³Àº ÁúÀÇ ¹®ÀÚ¿-ÀÇ ¹°À½Ç¥(?)·Î ¹Ù²Û´Ù. updates/inserts¸¦ À§ÇØ¼- ¿©·¯ºĐÀº ifx_update_blob(...)¸¦ »ç¿ëÇÏ¿© blobÀÇ ³»¿ëÀ» ¼³Á¤ÇÏ¿©¾ß ÇÑ´Ù.

BLOB Ä®·³¿¡ ´ëÇÑ µ¿ÀÛÀº ´ÙÀ½°ú °°Àº ¼³Á¤ º¯¼ö(configuration variables)¿¡ µû¶ó ´Þ¶óÁø´Ù. ÀÌ ¼³Á¤ º¯¼ö(configuration variables)Àº ½ÇÇàÁß¿¡µµ º¯°æÇÒ ¼ö ÀÖ´Ù. :

¼³Á¤º¯¼ö(configuration variable) :

ifx.textasvarchar

ifx.byteasvarchar

½ÇÇà ÇÔ¼ö(runtime functions) :

ifx_textasvarchar(0) : select ÁúÀÇ½Ã TEXT Ä®·³Ã³·³ blob id¸¦ »ç¿ëÇÑ´Ù.

ifx_byteasvarchar(0) : select ÁúÀÇ½Ã BYTE Ä®·³Ã³·³ blob id¸¦ »ç¿ëÇÑ´Ù.

ifx_textasvarchar(1) : select ÁúÀÇ¿¡¼- blob id¸¦ »ç¿ëÇÏÁö ¼Ê°í, VARCHAR Ä®·³ÀÎ °Í Ã³·³ TEXT Ä®·³À¸·Î ¹ÝÈ¯ÇÏ°í

ifx_byteasvarchar(1) : select ÁúÀÇ¿¡¼- blob id¸¦ »ç¿ëÇÏÁö ¼Ê°í, VARCHAR Ä®·³ÀÎ °Í Ã³·³ BYTE Ä®·³À¸·Î ¹ÝÈ¯ÇÏ°í

¼³Á¤º¯¼ö(configuration variable) :

ifx.blobinfile

½ÇÇà ÇÔ¼ö(runtime functions) :

ifx_blobinfile_mode(0) : ¸Þ¸ð¸®¿¡ ÀÖ´Â BYTE Ä®·³À» ¹ÝÈ¯ÇÑ´Ù. ¿©·¯ºÐÀº blob id¸¦ »ç¿ëÇÏ¿© ±× ³» ¿ëÀ» ¾òÀ» ¼ö ÀÖ´Ù.

ifx_blobinfile_mode(1) : ÆÄÀÏ¿¡ ÀÖ´Â BYTE Ä®·³À» ¹ÝÈ¯ÇÑ´Ù. ¿©·¯ºÐÀº blob id¸¦ »ç¿ëÇÏ¿© ±× ³»¿ëÀÌ ÀÖ´Â ÆÄÀÏÀÇ À̸§À» ¾òÀ» ¼ö ÀÖ´Ù.

¸¸¾à ¿©·¯ºÐÀÌ ifx_text/byteasvarchar¸¦ 1·Î ¼³Á¤ÇÏ¸é, ¿©·¯ºÐÀº select ÁúÀÇ¿¡¼- TEXT¿Í BYTE Ä®·³À» ÀÏ¹ÝÀûÀÎ (±×·¯³ª °¸´Ù ±ä) VARCHAR ÇÊµåÃ·³ »ç¿ëÇÒ ¼ö ÀÖ´Ù. PHP3¿¡¼- ¸ðµç ¹®ÀÚ¿-ÀÌ °è»ê("counted") µÇ¾îÁö´Â ÇÑ, ÀÌ°ÍÀº "binary safe"ÇÑ »óÅÂ·Î ÀÖ°Ô µÈ´Ù. ÀÌ°ÍÀ» ¿Å¹Ù¸£°Ô »ç¿ëÇÏ´Â °ÍÀº ¿©·¯ºÐ¿¡°Ô ´Þ·ÁÀÖ´Ù. µ¹·Á¹ÞÀ» µ¥ÀÌÅÍ´Â ¿©·¯ºÐÀÌ ±× ³»¿ë¿¡ ´ëÇØ Ã¥ÀÓÁú ¼ö ÀÖ´Ù¸é, ¾î¶² ³»¿ëµµ Æ÷ÇÔÇÒ ¼ö ÀÖ´Ù.

¸¸¾à ¿©·¯ºÐÀÌ ifx_blobinfileÀ» 1·Î ¼³Á¤ÇÏ¸é, blobÀÇ ³»¿ëÀ» °¡Á®¿ÀÅ© À§ÇØ ifx_get_blob(..)¸¦ »ç¿ëÇÏ¿© ¹ÝÈ¯ ¹ÞÀº ÆÄÀÏ¸íÀ» »ç¿ëÇÏ¿©¾ß ÇÑ´Ù. ÀÌ °æ¿ì¿¡ ¾î¶² row¸¦ °¡Á®¿À°Ô(fetch) µÇ¸é **¿©·¯ºÐÀº INFORMIX°¡ ¸¸µç ÀÓ½Ã ÆÄÀÏµéÀ» Áö¿öÁÙ Ã¥ÀÓÀÌ ÀÖ´Ù.** Informix´Â »õ row fetch¶§ ¸¶´Ù ¸ðµç BYTE Ä®·³¿¡ ÇÏ³ª¾¿ »õ ÀÓ½Ã ÆÄÀÏÀ» ¸¸µç´Ù.

ÀÓ½Ã ÆÄÀÏÀÌ ÀÖ´Â µð·º°Ä¸®´Â È¯°æ¯¼öÀÎ "blobdir"¿¡ ¼³Á¤µÈ µÈ °ªÀ» »ç¿ëÇÑ´Ù. ±â°»°ªÀº ÇöÀç µð·º°Ä¸®ÀÎ "." ÀÌ´Ù. putenv(blobdir=tmpblob"); °°ÀÌ »ç¿ëÇÏ¸é ³²°ÜÁø ("blb"·Î ½ÃÀÛÇÏ´Â ÀÌ¸§À» °¡Áø) ÀÓ½Ã ÆÄÀÏ µéÀ» ½±°Ô Ã»¼ÒÇÒ ¼ö ÀÖ´Ù.

ÀÚµ¿ °ø¹é Á¦°Å (Automatically trimming "char" <SQLCHAR and SQLNCHAR> data) :

ÀÌ ±â´É° ´ÙÀ½ ¼³Á¤ °¯¼ö(configuration variable)·Î ¼³Á¤ÇÒ ¼ö ÀÖ´Ù.

ifx.charasvarchar : 1·Î ¼³Á¤µÇ¾îÀÖ¸¸¸é µÚ¿¡ °ÙÀº °ø¹é¹®ÀÚ¸¦ ÀÚµ¿À¸·Î Á¦°ÅÇÑ´Ù.

---

ifx_connect

ifx_connect - - Informix ¼-¹ö connectionÀ» ¿¬´Ù.

Description

`int ifx_connect`(string [database] , string [userid] , string [password]);

Returns an connection identifier on success, or FALSE on error.

ifx_connect() establishes a connection to an Informix server. All of the arguments are optional, and if they're missing, defaults are taken from values supplied in php3.ini (ifx.default_host for the host (Informix libraries will use $INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

In case a second call is made to **ifx_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **ifx_close()**.

See also **ifx_pconnect()**, and **ifx_close()**.
Example 1. Connect to a Informix database

```
$conn_id = ifx_pconnect (mydb@ol_srv1, "imyself", "mypassword");
```

ifx_pconnect

ifx_pconnect - - ¿µ±¸ÀûÀÎ(persistent) Informix ¼¹ö connectionÀ» ¿¬°Ù.

Description

int ifx_pconnect(string [database] , string [userid] , string [password]);

Returns: A positive Informix persistent link identifier on success, or false on error

ifx_pconnect() acts very much like **ifx_connect()** with two major differences.

This function behaves exactly like **ifx_connect()** when PHP is not running as an Apache module. First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ifx_close()** will not close links established by **ifx_pconnect()**).

This type of links is therefore called 'persistent'.

See also: **ifx_connect()**.

ifx_close

ifx_close - - Informix connectionÀ» ´Ý´Â´Ù.

Description

int ifx_close(int [link_identifier]);

Returns: always true.

ifx_close() closes the link to an Informix database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by **ifx_pconnect()**.

See also: **ifx_connect()**, and **ifx_pconnect()**.
Example 1. Closing a Informix connection

```
$conn_id = ifx_connect (mydb@ol_srv, "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

ifx_query

ifx_query - - Informix ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

int ifx_query(string query, int [link_identifier] , int [cursor_type] , mixed [blobidarray]);

Returns: A positive Informix result identifier on success, or false on error.

An integer "result_id" used by other functions to retrieve the query results. Sets "affected_rows" for retrieval by the **ifx_affected_rows()** function.

ifx_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **ifx_connect()** was called, and use it.

Executes *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a mask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together. Non-select queries are "execute immediate".

For either query type the number of (estimated or real) affected rows is saved for retrieval by **ifx_affected_rows()**.

If you have BLOB (BYTE or TEXT) columns in an update query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: **ifx_connect()**.

Example 1. Show all rows of the "orders" table as a html table

```
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n",
        ifx_error();
        ifx_errormsg();
  die;
}
ifx_htmltbl_result($res_id, "border=\"1\");
ifx_free_result($res_id);
```

Example 2. Insert some values into the "catalog" table

```
                     // create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                     // store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                     // launch query
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
  ... error ...
}
                    // free result id
ifx_free_result($res_id);
```

ifx_prepare

ifx_prepare - - ½ÇÇàÀ» À§ÇÑ SQL¹®À» ÁØºñÇÑ´Ù.

Description

int ifx_prepare(string query, int conn_id, int [cursor_def], mixed blobidarray);

Returns a integer *result_id* for use by **ifx_do()**. Sets *affected_rows* for retrieval by the **ifx_affected_rows()** function.

Prepares *query* on connection *conn_id*. For "select - type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a mask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together.

For either query type the estimated number of affected rows is saved for retrieval by **ifx_affected_rows()**.

If you have BLOB (BYTE or TEXT) columns in the query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: **ifx_do()**.

ifx_do

ifx_do - - ÀÌÀü¿¡ ÁØºñµÈ SQL¹®À» ½ÇÇàÇÑ´Ù.

Description

int ifx_do(int result_id);

Returns TRUE on success, FALSE on error.

Executes a previously prepared query or opens a cursor for it.

Does NOT free *result_id* on error.

Also sets the real number of **ifx_affected_rows()** for non - select statements for retrieval by **ifx_affected_rows()**

See also: **ifx_prepare()**. There is a example.

ifx_error

ifx_error - - ¸¶Áö¸·¿¡ ¼öÇàÇÑ Informix ¸í·ÉÀÇ ¿¡·¯ ÄÚµå¸¦ µ¹·ÁÀ´Ù.

Description

string ifx_error(void);

The Informix error codes (SQLSTATE & SQLCODE) formatted as follows :

x [SQLSTATE = aa bbb SQLCODE=cccc]

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed. The format of the string is "(char) [SQLSTATE=(two digits) (three digits) SQLCODE=(one digit)]". The first character can be ' ' (space) (success), 'W' (the statement caused some warning), 'E' (an error happened when executing the statement) or 'N' (the statement didn't return any data).

See also: **ifx_errormsg()**

ifx_errormsg

ifx_errormsg - - ¸¶Áö¸·¿¡ ¼öÇàÇÑ Informix ¸í·ÉÀÇ ¿¡·¯ ¸Þ¼¼Áö¸¦ µ¹·ÁÀ´Ù.

Description

string ifx_errormsg(int [errorcode]);

Returns the Informix error message associated with the most recent Informix error, or, when the optional "*errorcode*" param is present, the error message corresponding to "*errorcode*".

See also: **ifx_error()**

```
printf("%s\n<br>", ifx_errormsg(-201));
```

ifx_affected_rows

ifx_affected_rows - - ÁúÀÇ¿¡ ¿µÇâÀ» ¹Þ´Â rowÀÇ °³¼ö¸¦ ±¸Ñ´Ù.

Description

int ifx_affected_rows(int result_id);

result_id is a valid result id returned by **ifx_query()** or **ifx_prepare()**.

Returns the number of rows affected by a query associated with *result_id*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it.

Useful after **ifx_prepare()** to limit queries to reasonable result sets.

See also: **ifx_num_rows()**
Example 1. Informix affected rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
```

ifx_getsqlca

ifx_getsqlca - - ÁúÀÇ ÈÄ sqlca.sqlerrd[0..5]ÀÇ ³»¿ëÀ» ±¸Ñ´Ù.

Description

array ifx_getsqlca(int result_id);

result_id is a valid result id returned by **ifx_query()** or **ifx_prepare()**.

Returns a pseudo-row (assiociative arry) with sqlca.sqlerrd[0] to sqlca.sqlerrd[5] after the query associated with *result_id*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For selects the values are those saved after the prepare statement. This gives access to the estimated number of affected rows. The use of this function saves the overhead of executing a "select dbinfo('sqlca.sqlerrdx')" query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.
Example 1. Retrieve Informix sqlca.sqlerrd[x] values

```
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable values(0, '2nd column', 'another column' ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n";
```

ifx_fetch_row

ifx_fetch_row - - row¸¦ ¹è¿(enumerated array)·Î °¡Á®¿Â´Ù.

Description

array ifx_fetch_row(int result_id, mixed [position]);

Returns an associative array that corresponds to the fetched row, or false if there are no more rows.

Blob columns are returned as integer blob id values for use in **ifx_get_blob()** unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values. Returns FALSE on error

result_id is a valid resultid returned by **ifx_query()** or **ifx_prepare()** (select type queries only!).

[position] is an optional parameter for a "fetch" operation on "scroll" cursors: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for scrollcursors.

ifx_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **ifx_fetch_row()** would return the next row in the result set, or false if there are no more rows.
Example 1. Informix fetch rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_htmltbl_result

ifx_htmltbl_result - - ÁúÀÇÀÇ ¸ðµç °á°ú row¸¦ HTML table ¸ð¾çÀ¸·Î Ãâ·ÂÇÑ´Ù.

Description

int ifx_htmltbl_result(int result_id, string [html_table_options]);

Returns the number of rows fetched or FALSE on error.

Formats all rows of the *result_id* query into a html table. The optional second argument is a string of <table> tag options
Example 1. Informix results as HTML table

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do($rid) {
  ... error ...
}
ifx_htmltbl_result ($rid, "border=\"2\"");
ifx_free_result($rid);
```

ifx_fieldtypes

ifx_fieldtypes - - Informix SQL ÇÊµåµéÀ» listÇÑ´Ù.

Description

array ifx_fieldtypes(int result_id);

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *result_id*. Returns FALSE on error.
Example 1. Fielnames and SQL fieldtypes

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
```

ifx_fieldproperties

ifx_fieldproperties - - SQL ÇÊµå Æ¯¼ºÀ» list ÇÑ´Ù.

Description

array ifx_fieldproperties(int result_id);

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *result_id*. Returns FALSE on error.

Returns the Informix SQL fieldproperies of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Example 1. Informix SQL fieldproperties

```
$properties = ifx_fieldtypes ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_num_fields

ifx_num_fields - - ÁúÀÇÀÇ ÄÃ·³ °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int ifx_num_fields(int result_id);

Returns the number of columns in query for *result_id* or FALSE on error

After preparing or executing a query, this call gives you the number of columns in the query.

ifx_num_rows

ifx_num_rows - - ÁúÀÇ °á°úÀÇ row °³¼ö

Description

int ifx_num_rows(int result_id);

Gives the number of rows fetched so far for a query with *result_id* after a **ifx_query()** or **ifx_do()** query.

ifx_free_result

ifx_free_result - - ÁúÀÇ¿¡ »ç¿ëµÈ ÀÚ¿øÀ» Ç®¾îÁØ´Ù.

Description

int ifx_free_result(int result_id);

Releases resources for the query associated with *result_id*. Returns FALSE on error.

ifx_create_char

ifx_create_char - - char object¸¦ ¸¸µç´Ù.

Description

int ifx_create_char(string param);

Creates an char object. *param* should be the char content.

ifx_free_char

ifx_free_char - - char object¸¦ »èÁ¦ÇÑ´Ù.

Description

int ifx_free_char(int bid);

Deletes the charobject for the given char object - id *bid*. Returns FALSE on error otherwise TRUE.

ifx_update_char

ifx_update_char - - char objectÀÇ ³»¿ëÀ» ¼öÁ¤ÇÑ´Ù.

Description

int ifx_update_char(int bid, string content);

Updates the content of the char object for the given char object *bid*. *content* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_get_char

ifx_get_char - - char objectÀÇ ³»¿ëÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

int ifx_get_char(int bid);

Returns the content of the char object for the given char object - id *bid*.

ifx_create_blob

ifx_create_blob - - blob object¸¦ ¸¸µç´Ù.

Description

int ifx_create_blob(int type, int mode, string param);

Creates an blob object.

type: 1 = TEXT, 0 = BYTE

mode: 0 = blob - object holds the content in memory, 1 = blob - object holds the content in file.

param: if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.

Return FALSE on error, otherwise the new blob object - id.

ifx_copy_blob

ifx_copy_blob - - ÁÕ¼îÁø blob objectÀÇ »ç»»À» ¸¸µç´Ù.

Description

int ifx_copy_blob(int bid);

Duplicates the given blob object. *bid* is the ID of the blob object.

Returns FALSE on error otherwise the new blob object - id.

ifx_free_blob

ifx_free_blob - - blob object¸¦ »èÁ¦ÇÑ´Ù.

Description

int ifx_free_blob(int bid);

Deletes the blobobject for the given blob object - id *bid*. Returns FALSE on error otherwise TRUE.

ifx_get_blob

ifx_get_blob - - blob objectÀÇ ³»¿ëÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

int ifx_get_blob(int bid);

Returns the content of the blob object for the given blob object - id *bid*.

ifx_update_blob

ifx_update_blob - - blob objectÀÇ ³»¿ëÀ» ¼öÁ¤ÇÑ´Ù.

Description

ifx_update_blob(int bid, string content);

Updates the content of the blob object for the given blob object *bid*. *content* is a string with new data. Returns FALSE on error otherwise TRUE.

ifx_blobinfile_mode

ifx_blobinfile_mode - - select ÁúÀÇ¿¡¼- »ç¿ëµÇ´Â ±â°»(default) blob mode¸¦ ¼³Á¤ÇÑ´Ù.

Description

void ifx_blobinfile_mode(int mode);

Set the default blob mode for all select queries. Mode "0" means save Byte - Blobs in memory, and mode "1" means save Byte - Blobs in a file.

ifx_textasvarchar

ifx_textasvarchar - - select ÁúÀÇ¿¡¼- »ç¿ëµÇ´Â ±â°»(default) text mode¸¦ ¼³Á¤ÇÑ´Ù.

Description

void ifx_textasvarchar(int mode);

Sets the default text mode for all select - queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_byteasvarchar

ifx_byteasvarchar - - select ÁúÀÇ¿¡¼- »ç¿ëµÇ´Â ±â°»(default) byte mode¸¦ ¼³Á¤ÇÑ´Ù.

Description

void ifx_byteasvarchar(int mode);

Sets the default byte mode for all select - queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_nullformat

ifx_nullformat - - row¸¦ °¡Á®¿Ã ¶§ »ç¿ëÇÒ °á°ú°ªÀÇ ±â°» ÇüÅÂ¸¦ Á¤ÇÑ´Ù.

Description

void ifx_nullformat(int mode);

Sets the default return value of a NULL-value on a fetch row. Mode "0" returns "", and mode "1" returns "NULL".

ifxus_create_slob

ifxus_create_slob - - slob object¸¦ ¸¸µé°í, ±×°ÍÀ» ¿¬´Ù.

Description

int ifxus_create_slob(int mode);

Creates an slob object and opens it. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER - > or-mask. You can also use constants named IFX_LO_RDONLY, IFX_LO_WRONLY etc. Return FALSE on error otherwise the new slob object-id.

ifx_free_slob

ifx_free_slob - - slob object¸¦ »èÁ¦ÇÑ´Ù.

Description

int ifxus_free_slob(int bid);

Deletes the slob object. *bid* is the Id of the slob object. Returns FALSE on error otherwise TRUE.

ifxus_close_slob

ifxus_close_slob - - slob object¸¦ »èÁ¦ÇÑ´Ù.

Description

int ifxus_close_slob(int bid);

Deletes the slob object on the given slob object-id *bid*. Return FALSE on error otherwise TRUE.

ifxus_open_slob

ifxus_open_slob - - slob object¸¦ ¿¬´Ù.

Description

int ifxus_open_slob(long bid, int mode);

Opens an slob object. *bid* should be an existing slob id. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER - > or-mask. Returns FALSE on error otherwise the new slob object-id.

ifxus_tell_slob

ifxus_tell_slob - - ÇöÀç ÆÄÀÏÀÌ³ª seek À§Ä¡¸¦ ¹Ý¯ÇÑ´Ù.

Description

int ifxus_tell_slob(long bid);

Returns the current file or seek position of an open slob object *bid* should be an existing slob id. Return FALSE on error otherwise the seek position.

ifxus_seek_slob

ifxus_seek_slob - - ÇöÀç ÆÄÀÏÀÌ³ª seek À§Ä¡¸¦ ¼³Á¤ÇÑ´Ù.

Description

int ifxus_seek_blob(long bid, int mode, long offset);

Sets the current file or seek position of an open slob object. *bid* should be an existing slob id. Modes: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END and *offset* is an byte offset. Return FALSE on error otherwise the seek position.

ifxus_read_slob

ifxus_read_slob - - slob objectÀÇ nbytes¸¦ ÀÐ´Â´Ù.of the

Description

int ifxus_read_slob(long bid, long nbytes);

Reads nbytes of the slob object. *bid* is a existing slob id and *nbytes* is the number of bytes zu read. Return FALSE on error otherwise the string.

ifxus_write_slob

ifxus_write_slob - - slob object¿¡ ¹®ÀÚ¿-À» ½á ³Ö´Â´Ù.

Description

int ifxus_write_slob(long bid, string content);

Writes a string into the slob object. *bid* is a existing slob id and *content* the content to write. Return FALSE on error otherwise bytes written.

XXII. InterBase Functions

Table of Contents

ibase_connect

ibase_connect - -

Description

ibase_connect( );

ibase_pconnect

ibase_pconnect - -

Description

ibase_pconnect( );

ibase_close

ibase_close - -

Description

ibase_close( );

ibase_query

ibase_query - -

Description

ibase_query( );

ibase_fetch_row

ibase_fetch_row - -

Description

ibase_fetch_row( );

ibase_free_result

ibase_free_result - -

Description

ibase_free_result( );

ibase_prepare

ibase_prepare - -

Description

ibase_prepare( );

ibase_bind

ibase_bind - -

Description

ibase_bind( );

ibase_execute

ibase_execute - -

Description

ibase_execute( );

ibase_free_query

ibase_free_query - -

Description

ibase_free_query( );

ibase_timefmt

ibase_timefmt - -

Description

```
ibase_timefmt( );
```

XXIII. LDAP Functions

Table of Contents

Introduction to LDAP

LDAPÀº Lightweight Directory Access ProtocolÀÇ ¾àÀÚ·Î, "Directory Servers"¿¡ Á¢±ÙÇÏ´Âµ¥ »ç¿ëµÇ´Â ÇÁ·ÎÅäÄÝÀÌ´Ù. Directory ¶ó´Â °ÍÀº Æ®¸®Á¶¸¦ °¡Áö°í Á¤º¸¸¦ ÀúÀåÇÏ´Â Æ¯°°ÇÑ Á¾·ùÀÇ Database¶ó°í ÇÒ ¼ö ÀÖ´Ù.

±â°» °³³äÀº ÇÏµåµð½º¶©ÀÇ µð·ºÅä¸® ±¸Á¶¿Í ºñ½ÁÇÏ´Ù. ´Ù¸¥ Á¡À° root µð·ºÅä¸®°¡ "world(Áö±¸ ÀüÃ¼)"ÀÌ°í, Ã¹ ¹øÂ° ·¹º§ÀÌ "±¹°¡µé", ±× ÀÌÇÏÀÇ ·¹º§ÀÌ È¸»ç³ª, °¢Á¾ Á¶Á÷, Àå¼Ò µîÀÌ µÈ´Ù´Â Á¡ÀÌ´Ù. ±× ÀÌÇÏ·Î °è¼Ó ³»·Á°¡´Ù °¸¸é Æ¯Á¤ÀÎÀÌ³ª ¼³°ñ, ¹®¼ µîÀÇ Ç×¸ñ±îÁöµµ ³»·Á°¥ ¼ö ÀÖ´Ù.

ÇÏµåµð½º¶©ÀÇ subdirectory¿¡ ÀÖ´Â ÆÄÀÏÀ» ÂüÁ¶ÇÒ ¶§ °¸Åë ´ÙÀ½°ú °°ÀÌ »ç¿ëÇÑ´Ù. :

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

ÀÌ ÆÄÀÏ ·¹ÆÛ·±½º¶ÀÇ °¢°Î°ÐÀº ½½·¡½Ã(/) ±âÈ£·Î ±¸ÐµÇ°í, ÀÐ´Â ¼ø¼´Â ¿ÞÂÊ¿¡¼ ¿À¸¥ÂÊÀ¸·Î ÀÐ¾î³ª°£´Ù.

LDAP¿¡¼ ÆÄÀÏ ·¹ÆÛ·±½º¿Í °°À° °ÍÀº °¸Åë "dn"À¸·Î Ç¥½ÃµÇ´Â "distinguished name"ÀÌ´Ù. dnÀÇ ¿¹´Â ´ÙÀ½°ú °°´Ù. :

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

·¹ÆÛ·±½º¶ÀÇ °¢ °Î°ÐÀº ½°Ç¥(,)·Î ±¸ÐµÇ°í, ¿À¸¥ÂÊ¿¡¼ ¿ÞÂÊÀÇ ¼ø¼·Î ÀÐ´Â´Ù. ¿©·¯°ÐÀº ÀÌ dnÀ» ´ÙÀ½°ú °°À° ÀÇ¹Ì·Î ÀÐ ´Â´Ù. :

```
country(³ª¶ó) = US
organization(Á¶Á÷) = My Company
organizationalUnit(°Î¼) = Accounts
commonName(ÀÌ¸§) = John Smith
```

ÇÏµåµð½º¿¡¼ ¿©·¯°ÐÀÌ µð·ºÅä¸® ±¸¼º°À» ¾î¶»°Ô ÇÏ´Â°¡¿¡ ´ëÇÑ Æ¯°°ÇÑ ±ÔÄ¢ÀÌ ¼øµíÀÌ, µð·ºÅä¸® ¼¹öÀÇ °ü¸®µµ ¸ñÀû¿¡ µû¶ó ÀÇ¹ÌÀÖ°Ô ±¸¼ºµÈ´Ù´é ¾î¶² ð½ÀÀ» ÇÏ°í ÀÖ¾îµµ »ó¹ü¾ø´Ù. ÇÏÁö¸¸, »ç¿ëµÇ°í ÀÖ´Â ¾à°£ÀÇ °ü½ÀÀÌ ÀÖ´Ù. ±× ¸ñ½ÃÁö´Â ¿©·¯°ÐÀÌ µ¥ÀÌÅ¸°£ÀÌ½º¿¡ ¾î¶² Á¤º¸¸¦ »ç¿ë°¡´ÉÇÑÁö ¼ËÁö ¸øÇÑ´Ù´é ±× µ¥ÀÌÅÍ°£ÀÌ½º¸¦ »ç¿ëÇÒ ¼ö ¾øµíÀÌ, ¿©·¯°ÐÀÌ µð·ºÅä¸® ±¸¶¿¡ ´ëÇØ ¾ËÁö ¸øÇÑ´Ù´é ¿©·¯°ÐÀº ±× µð·ºÅä¸® ¼¹öÀÇ ¿ÛÇ±ÙÇÏ±â À§ÇÑ ÄÚµå¸¦ ¼ø¾øÀÌ ´Ù½Ã °ÍÀÌ´Ù.

¿¹Á¦ (Complete code example)

µð·°Åä¸® ¼-¹ö¿¡¼- "S"·Î ½ÃÀÛÇÏ´Â ¼°À» °¡Áø Ç×¸ñ¿¡ ´ëÇÑ Á¤º¸¸¦ °Ë»öÇÏ°í, ÀÌ¸§°ú email ÁÖ¼Ò¸¦ Ãâ·ÂÇÑ´Ù.
Example 1. LDAP search example

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection
echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // must be a valid LDAP server!
echo "connect result is ".$ds."<p>";
if ($ds) {
    echo "Binding ...";
    $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
                           // read-only access echo "Bind result is
    echo "Bind result is ".$r."<p>";
    echo "Searching for (sn=S*) ...";
    // Search surname entry
    $sr=ldap_search($ds,"o=My Company, c=US", "sn=S*");
    echo "Search result is ".$sr."<p>";
    echo "Number of entires returned is ".ldap_count_entries($ds,$sr)."<p>";
    echo "Getting entries ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for ".$info["count"]." items returned:<p>";
    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn is: ". $info[$i]["dn"] ."<br>";
        echo "first cn entry is: ". $info[$i]["cn"][0] ."<br>";
        echo "first email entry is: ". $info[$i]["mail"][0] ."<p>";
    }
    echo "Closing connection";
    ldap_close($ds);
} else {
    echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

PHP¿¡¼- LDAP È£Ãâ »ç¿ëÇÏ±â (Using the PHP LDAP calls)

¿©·¯°ÐÀº University of Michigan ldap - 3.3 package³ª Netscape Directory SDK°°Àº LDAP client libraries¸¦ ¾ò¾î¼- ÄÄÆÄÀÏÇÒ ÇÊ¿äÇÏ ÀÖ´Ù. ¶ÇÇÑ PHP¸¦ LDAPÁö¿øÀ¸·Î ´Ù½Ã ÄÄÆÄÀÏÇÒ ÇÊ¿ä° ÀÖ´Ù.

LDAP È£ÃâÀ» »ç¿ëÇÏ±â Àü¡ ¿©·¯°ÐÀº ´ÙÀ½°ú °°Àº °ÍÀ» ¾Ë°í ÀÖ¾î¾ß ÇÑ´Ù.

¿©·¯°ÐÀÌ »ç¿ëÇÏ·ÁÇÏ´Â LDAP ¼-¹öÀÇ ÀÌ¸§ÀÌ³ª ÁÖ¼Ò.

ÇØ´ç ¼-¹öÀÇ "base dn" (world µð·°Åä¸® ¹ØÀÇ ¾î¶² ÇÑ °Î°ÐÀ¸·Î ÇØ´ç ¼-¹ö¿¡ ÀúÀåµÈ °Î°Ð. "o=My Company, c=US"°°Àº °ÍÀÌ °¡´ÉÇÏ´Ù.)

¼-¹ö¿¡ Á¢±Ù°¡´ÉÇÑ ¾ÏÈ£°¡ ÇÊ¿äÇÒ ¼öµµ ÀÖ´Ù. (¸¹Àº ¼-¹öµéÀÌ "anonymous bind"¸¦ »ç¿ëÇ¿© ÀÐ±â Á¢±ÙÀ» Çã¿ëÇÏ°í ÀÖÁö¸¸ ±×¿ÜÀÇ µ¿ÀÛÀ» À§ÇØ¼-´Â ¾ÏÈ£¸¦ ¿ä±¸Çß´Ù.)

¿©·¯°ÐÀÌ applicationÀ» ÀÛ¼ºÇÒ ¶§ »ç¿ëÇÒ ÀüÇüÀûÀÎ LDAP È£ÅâÀº ´ÙÀ½ÀÇ Çü½ÄÀ» µû¸¥´Ù. :

```
ldap_connect()    // ¼-¹ö¿¡ connectionÀ» ¸¸µç´Ù.
   |
ldap_bind()       // ÀÍ¸í(anonymous)ÀÌ³ª ÀÎÁõµÈ "login"
   |
µð·°Åä¸®ÀÇ °Ë»öÀÌ³ª ¼öÁ¤ °°Àº ÀÛ¾÷À» ÇÏ°í °á°ú¸¦ Ãâ·ÂÇÑ´Ù.
   |
ldap_close()      // "logout"
```

More Information

LDAP¿¡ °üÇÑ ÀÚ¼¼ÇÑ ÀÚ·á´Â ´ÙÀ½ »çÀÌÆ®¿¡ °¡¸¸ÀÚ.

Netscape

University of Michigan

OpenLDAP Project

LDAP World

Netscape SDK¿¡´Â Programmer's Guide°¡ .html ÇüÅÂ·Î ÀÖ½À´Ï´Ù.

---

ldap_add

ldap_add - - LDAP directory¿¡ entry¸¦ Ãß°¡ÇÑ´Ù.

Description

`int ldap_add`(int link_identifier, string dn, array entry);

returns true on success and false on error.

The **ldap_add()** function is used to add entries in the LDAP directory. The DN of the entry to be added is specified by dn. Array entry specifies the information about the entry. The values in the entries are indexed by individual attributes. In case of multiple values for an attribute, they are indexed using integers starting with 0.

```
entry["attribute1"] = value
entry["attribute2"][0] = value1
entry["attribute2"][1] = value2
```

Example 1. Complete example with authenticated bind

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host
if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds,"cn=root, o=My Company, c=US", "secret");
    // prepare data
    $info["cn"]="John Jones";
    $info["sn"]="Jones";
    $info["mail"]="jonj@here.and.now";
    $info["objectclass"]="person";
    // add data to directory
    $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);
    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
}
?>
```

ldap_mod_add

ldap_mod_add - - ÇöÀç ¼Ó¼º¿¡ Æ¯Á¤ ¼Ó¼º°ªÀ» ´õÇÑ´Ù.

Description

`int ldap_mod_add`(int link_identifier, string dn, array entry);

returns true on success and false on error.

This function adds attribute(s) to the specified dn. It performs the modification at the attribute level as opposed to the object level. Object - level additions are done by the **ldap_add()** function.

ldap_mod_del

ldap_mod_del - - ÇöÀç ¼Ó¼º¿¡ Æ¯Á¤ ¼Ó¼º°ªÀ» »«´Ù.

Description

`int ldap_mod_del`(int link_identifier, string dn, array entry);

returns true on success and false on error.

This function removes attribute(s) from the specified dn. It performs the modification at the attribute level as opposed to the object level. Object - level deletions are done by the **ldap_del()** function.

ldap_mod_replace

ldap_mod_replace - - ÇöÀç ¼Ó¼º°ª µéÀ» »õ·Î¿î °ªÀ¸·Î ¹Ù²Û´Ù.

Description

`int ldap_mod_replace`(int link_identifier, string dn, array entry);

returns true on success and false on error.

This function replaces attribute(s) from the specified dn. It performs the modification at the attribute level as opposed to the object level. Object-level modifications are done by the **ldap_modify()** function.

ldap_bind

ldap_bind - - LDAP directory¿¡ bind ÇÑ´Ù.

Description

int ldap_bind(int link_identifier, string bind_rdn, string bind_password);

Binds to the LDAP directory with specified RDN and password. Returns true on success and false on error.

ldap_bind() does a bind operation on the directory. bind_rdn and bind_password are optional. If not specified, anonymous bind is attempted.

ldap_close

ldap_close - - LDAP server¿ÍÀÇ ¿¬°áÀ» ´Ý´Â´Ù.

Description

`int ldap_close`(int link_identifier);

Returns true on success, false on error.

ldap_close() closes the link to the LDAP server that's associated with the specified *link_identifier*.

This call is internally identical to **ldap_unbind()**. The LDAP API uses the call **ldap_unbind()**, so perhaps you should use this in preference to **ldap_close()**.

ldap_connect

ldap_connect - - LDAP server¿Í ¿¬°áÇÑ´Ù.

Description

int ldap_connect(string hostname, int port);

Returns a positive LDAP link identifier on success, or false on error.

ldap_connect() establishes a connection to a LDAP server on a specified *hostname* and *port*. Both the arguments are optional. If no arguments are specified then the link identifier of the already opened link will be returned. If only *hostname* is specified, then the port defaults to 389.

ldap_count_entries

ldap_count_entries - - Å½»ö ¹üÀ§ ¼È¿¡ ÀÖ´Â entryÀÇ °³¼ö

Description

int ldap_count_entries(int link_identifier, int result_identifier);

Returns number of entries in the result or false on error.

ldap_count_entries() returns the number of entries stored in the result of previous search operations. *result_identifier* identifies the internal ldap result.

ldap_delete

ldap_delete - - directory¾ÈÀÇ ÇÑ entry¸¦ »èÁ¦ÇÑ´Ù..

Description

int ldap_delete(int link_identifier, string dn);

Returns true on success and false on error.

ldap_delete() function delete a particular entry in LDAP directory specified by dn.

ldap_dn2ufn

ldap_dn2ufn - - DNÀ» User Friendly Naming Çü½ÄÀ¸·Î º¯È¯ÇÑ´Ù.

Description

string ldap_dn2ufn(string dn);

ldap_dn2ufn() function is used to turn a DN into a more user-friendly form, stripping off type names.

ldap_explode_dn

ldap_explode_dn - - DNÀ» ±×°ÍÀÇ ±¸¼º ¿ÎºÐÀ¸·Î ³ª´«´Ù.

Description

string ldap_explode_dn(string dn, int with_attrib);

ldap_explode_dn() function is used to split the a DN returned by **ldap_get_dn()** and breaks it up into its component parts. Each part is known as Relative Distinguished Name, or RDN. **ldap_explode_dn()** returns an array of all those components. *with_attrib* is used to request if the RDNs are returned with only values or their attributes as well. To get RDNs with the attributes (i.e. in attribute=value format) set *with_attrib* to 1 and to get only values set it to 0.

ldap_first_attribute

ldap_first_attribute - - Ã¹øÂ° ¼Ó¼º°À» ¹ÝÈ¯ÇÑ´Ù.

Description

string ldap_first_attribute(int link_identifier, int result_entry_identifier, int ber_identifier);

Returns the first attribute in the entry on success and false on error.

Similar to reading entries, attributes are also read one by one from a particular entry. **ldap_first_attribute()** returns the first attribute in the entry pointed by the entry identifier. Remaining attributes are retrieved by calling **ldap_next_attribute()** successively. *ber_identifier* is the identifier to internal memory location pointer. It is passed by reference. The same *ber_identifier* is passed to the ldap_next_attribute() function, which modifies that pointer.

see also **ldap_get_attributes()**

ldap_first_entry

ldap_first_entry - - Ã¹øÂ° result id¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int ldap_first_entry(int link_identifier, int result_identifier);

Returns the result entry identifier for the first entry on success and false on error.

Entries in the LDAP result are read sequentially using the **ldap_first_entry()** and **ldap_next_entry()** functions. **ldap_first_entry()** returns the entry identifier for first entry in the result. This entry identifier is then supplied to **lap_next_entry()** routine to get successive entries from the result.

see also **ldap_get_entries()**.

ldap_free_result

ldap_free_result - - result memory¸¦ Ç®¾îÁØ´Ù.

Description

int ldap_free_result(int result_identifier);

Returns true on success and false on error.

ldap_free_result() frees up the memory allocated internally to store the result and pointed by the *result_identifier*. All result memory will be automatically freed when the script terminates.

Typically all the memory allocated for the ldap result gets freed at the end of the script. In case the script is making successive searches which return large result sets, **ldap_free_result()** could be called to keep the runtime memory usage by the script low.

ldap_get_attributes

ldap_get_attributes - - search result entry·Î°ÎÅÍ ¼Ó¼º¾À» ±¸ÇÑ´Ù.

Description

array ldap_get_attributes(int link_identifier, int result_entry_identifier);

Returns a comlete entry information in a multi-dimensional array on success and false on error.

ldap_get_attributes() function is used to simplify reading the attributes and values from an entry in the search result. The return value is a multi-dimensional array of attributes and values.

Having located a specific entry in the directory, you can find out what information is held for that entry by using this call. You would use this call for an application which "browses" directory entries and/or where you do not know the structure of the directory entries. In many applications you will be searching for a specific attribute such as an email address or a surname, and won't care what other data is held.

return_value["count"] = number of attributes in the entry
return_value[0] = first attribute
return_value[n] = nth attribute

return_value["attribute"]["count"] = number of values for attribute
return_value["attribute"][0] = first value of the attribute
return_value["attribute"][i] = ith value of the attribute

Example 1. Show the list of attributes held for a particular directory entry

```
// $ds is the link identifier for the directory
// $sr is a valid search result from a prior call to
// one of the ldap directory search calls
$entry = ldap_first_entry($ds, $sr);
$attrs = ldap_get_attributes($ds, $entry);
echo $attrs["count"]." attributes held for this entry:<p>";
for ($i=0; $i<$attrs["count"]; $i++)
    echo $attrs[$i]."<br>";
```

see also **ldap_first_attribute()** and **ldap_next_attribute()**

ldap_get_dn

ldap_get_dn - - result entryÀÇ DNÀ» ±¸ÇÑ´Ù.

Description

string ldap_get_dn(int link_identifier, int result_entry_identifier);

Returns the DN of the result entry and false on error.

ldap_get_dn() function is used to find out the DN of an entry in the result.

ldap_get_entries

ldap_get_entries - - ¸ðµç result entryµéÀ» ±¸ÇÑ´Ù.

Description

array ldap_get_entries(int link_identifier, int result_identifier);

Returns a complete result information in a multi - dimenasional array on success and false on error.

ldap_get_entries() function is used to simplify reading multiple entries from the result and then reading the attributes and multiple values. The entire information is returned by one function call in a multi - dimensional array. The structure of the array is as follows.

The attribute index is converted to lowercase. (Attributes are case - insensitive for directory servers, but not when used as array indices)

return_value["count"] = number of entries in the result
return_value[0] : refers to the details of first entry

return_value[i]["dn"] = DN of the ith entry in the result

return_value[i]["count"] = number of attributes in ith entry
return_value[i][j] = jth attribute in the ith entry in the result

return_value[i]["attribute"]["count"] = number of values for attribute in ith entry
return_value[i]["attribute"][j] = jth value of attribute in ith entry

see also **ldap_first_entry()** and **ldap_next_entry()**

ldap_get_values

ldap_get_values - - result entry·Î ºÎÅÍ ¸ðµç °ªÀ» ±¸ÇÑ´Ù.

Description

array ldap_get_values(int link_identifier, int result_entry_identifier, string attribute);

Returns an array of values for the attribute on success and false on error.

ldap_get_values() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This call needs a *result_entry_identifier*, so needs to be preceded by one of the ldap search calls and one of the calls to get an individual entry.

You application will either be hard coded to look for certain attributes (such as "surname" or "mail") or you will have to use the **ldap_get_attributes()** call to work out what attributes exist for a given entry.

LDAP allows more than one entry for an attribute, so it can, for example, store a number of email addresses for one person's directory entry all labeled with the attribute "mail"

return_value["count"] = number of values for attribute
return_value[0] = first value of attribute
return_value[i] = ith value of attribute

Example 1. List all values of the "mail" attribute for a directory entry

```
// $ds is a valid link identifier for a directory server
// $sr is a valid search result from a prior call to
//     one of the ldap directory search calls
// $entry is a valid entry identifier from a prior call to
//        one of the calls that returns a directory entry
$values = ldap_get_values($ds, $entry,"mail");
echo $values["count"]." email addresses for this entry.<p>";
for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
```

ldap_list

ldap_list - - Single - level °Ë»ö (search)

Description

`int ldap_list`(int link_identifier, string base_dn, string filter);

Returns a search result identifier or false on error.

ldap_list() performs the search for a specified filter on the directory with the scope LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL means that the search should only return information that is at the level immediately below the base dn given in the call. (Equivalent to typing "ls" and getting a list of files and folders in the current working directory.)

This call takes an optional fourth parameter which is an array of the attributes required. See **ldap_search()** notes.
Example 1. Produce a list of all organizational units of an organization

```
// $ds is a valid link identifier for a directory server
$basedn = "o=My Company, c=US";
$justthese = array("ou");
$sr=ldap_list($ds, $basedn, "ou=*", $justthese);
$info = ldap_get_entries($ds, $sr);
for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

ldap_modify

ldap_modify - - LDAP entry¸¦ ¼öÁ¤ÇÑ´Ù.

Description

int ldap_modify(int link_identifier, string dn, array entry);

Returns true on success and false on error.

ldap_modify() function is used to modify the existing entries in the LDAP directory. The structure of the entry is same as in **ldap_add()**.

ldap_next_attribute

ldap_next_attribute - - resultÀÇ ´ÙÀ½ ¼Ó¼º» ±¸ÇÑ´Ù.

Description

`string ldap_next_attribute`(int link_identifier, int result_entry_identifier, int ber_identifier);

Returns the next attribute in an entry on success and false on error.

ldap_next_attribute() is called to retrieve the attributes in an entry. The internal state of the pointer is maintained by the *ber_identifier*. It is passed by reference to the function. The first call to **ldap_next_attribute()** is made with the *result_entry_identifier* returned from **ldap_first_attribute()**.

see also **ldap_get_attributes()**

ldap_next_entry

ldap_next_entry - - ´ÙÀ½ result entry¸¦ ±¸ÇÑ´Ù.

Description

`int ldap_next_entry`(int link_identifier, int result_entry_identifier);

Returns entry identifier for the next entry in the result whose entries are being read starting with **ldap_first_entry()**. If there are no more entries in the result then it returns false.

ldap_next_entry() function is used to retrieve the entries stored in the result. Successive calls to the **ldap_next_entry()** return entries one by one till there are no more entries. The first call to **ldap_next_entry()** is made after the call to **ldap_first_entry()** with the result_identifier as returned from the **ldap_first_entry()**.

see also **ldap_get_entries()**

ldap_read

ldap_read - - entry¸¦ ÀÐ´Â´Ù.

Description

int ldap_read(int link_identifier, string base_dn, string filter, array [attributes]);

Returns a search result identifier or false on error.

ldap_read() performs the search for a specified filter on the directory with the scope LDAP_SCOPE_BASE. So it is equivalent to reading an entry from the directory.

An empty filter is not allowed. If you want to retrieve absolutely all information for this entry, use a filter of "objectClass=*". If you know which entry types are used on the directory server, you might use an appropriate filter such as "objectClass=inetOrgPerson".

This call takes an optional fourth parameter which is an array of the attributes required. See **ldap_search()** notes.

ldap_search

ldap_search - - LDAP tree¸¦ °Ë»öÇÑ´Ù.

Description

int ldap_search(int link_identifier, string base_dn, string filter, array [attributes]);

Returns a search result identifier or false on error.

ldap_search() performs the search for a specified filter on the directory with the scope of LDAP_SCOPE_SUBTREE. This is equivalent to searching the entire directory. *base_dn* specifies the base DN for the directory.

There is a optional fourth parameter, that can be added to restrict the attributes and values returned by the server to just those required. This is much more efficient than the default action (which is to return all attributes and their associated values). The use of the fourth parameter should therefore be considered good practice.

The fourth parameter is a standard PHP string array of the required attributes, eg array("mail","sn","cn") Note that the "dn" is always returned irrespective of which attributes types are requested.

Note too that some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set.

The search filter can be simple or advanced, using boolean operators in the format described in the LDAP doumentation (see the Netscape Directory SDK for full information on filters).

The example below retrieves the organizational unit, surname, given name and email address for all people in "My Company" where the surname or given name contains the substring $person. This example uses a boolean filter to tell the server to look for information in more than one attribute.
Example 1. LDAP search

```
// $ds is a valid link identifier for a directory server
// $person is all or part of a person's name, eg "Jo"
$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array( "ou", "sn", "givenname", "mail");
$sr=ldap_search($ds, $dn, $filter, $justthese);
$info = ldap_get_entries($ds, $sr);
print $info["count"]." entries returned<p>";
```

When you perform a search, and too much data comes back (alot of entries) you will get a warning, and **ldap_get_entries()** will fail. The trick here is to turn off the warnings, then check the return value.

```
$normerr = error_reporting ();
error_reporting (0);  // turn off warnings!
$sr = ldap_search ($ds, $dn, $searchfor);
$normerr = error_reporting ($normerr);
if (!$sr) {
    print "too many entries!";
} else .....
```

You could try narrowing the scope, by adding an extra filter eg. (cn=a*), but It would be nicer to be able to grab the results in bits (eg. 1 - 100, 101 - 200...).

ldap_unbind

ldap_unbind - - LDAP directory·Î°ÎÅÍ unbindÇÑ´Ù.

Description

int ldap_unbind(int link_identifier);

Returns true on success and false on error.

ldap_unbind() function unbinds from the LDAP directory.

XXIV. Mail Functions

Table of Contents

mail() ÇÔ¼ö´Â ÆíÁö¸¦ º¸³¾ ¼ö ÀÖ°Ô ÇØ ÁØ´Ù.

mail

mail - - ÆíÁö¸¦ º¸³½´Ù.

Description

bool mail(string to, string subject, string message, string additional_headers);

Mail() automatically mails the message specified in *message* to the receiver specified in *to*. Multiple recipients can be specified by putting a space between each address in *to*.
Example 1. Sending mail.

```
mail("rasmus@lerdorf.on.ca", "My Subject", "Line 1\nLine 2\nLine 3");
```

If a fourth string argument is passed, this string is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline.
Example 2. Sending mail with extra headers.

```
mail("ssb@guardian.no", "the subject", $message,
     "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-Mailer: PHP/" . phpversion());
```

(¿ªÀÚÁÖ : ´ÙÀ½°ú °°ÀÌ \r\nÀ» »ç¿ëÇÏ¿© header¿¡ ¿©·¯ ³»¿ëÀ» Ã·°¡ÇÒ ¼ö ÀÖ´Ù.
```
mail("ssb@guardian.no", "the subject", $message, "X-Mailer: PHP/" . phpversion()."\r\nFrom:webmaster@yourhost.com\r\nReply-To:
webmaster@yourhost.com");
```
)

XXV. Mathematical Functions

Table of Contents

Introduction

ÀÌ ¼öÇÐ ÇÔ¼öµéÀº ¿©·¯°ÐÀÇ ÄÄÇ»ÅÍÀÇ long°ú double ÇüÅÂ·Î ¸»¿ëµÈ´Ï´Ù. ¸¸¾à ¿©·¯°ÐÀÌ ÀÌ ÀÌ»óÀÇ ¼ýÀÚ¸¦ ´Ù·ç¾î ¾ßÇÑ´Ù¸é arbitrary precision math functions¸¦ »ç¿ëÇÏµµ·ÏÇÑ´Ù.

Math constants

PHPÀÇ ¼öÇÐ ÇÔ¼ö È®Àå¿¡¼ ´ÙÀ½°ªÀÌ »ó¼ö·Î ÁöÁ¤µÇ¾îÀÖ´Ù. :

Table 1. Math constants

Constant	Value	Description
M_PI	3.14159265358979323846	The value of ¥ð(pi)

Abs

Abs - - Àý´ë°ª

Description

mixed abs(mixed number);

Returns the absolute value of number. If the argument number is float, return type is also float, otherwise it is int.

Acos

Acos - - arc cosine

Description

float acos(float arg);

Returns the arc cosine of arg in radians.

See also **asin()** and **atan()**.

Asin

Asin - - arc sine

Description

float asin(float arg);

Returns the arc sine of arg in radians.

See also **acos()** and **atan()**.

Atan

Atan - - arc tangent

Description

float atan(float arg);

Returns the arc tangent of arg in radians.

See also **acos()** and **atan()**.

Atan2

Atan2 - - arc tangent of two variables

Description

float atan2(float y, float x);

This function calculates the arc tangent of the two variables x and y. It is similar to calculating the arc tangent of y / x, except that the signs of both arguments are used to determine the quadrant of the result.

The function returns the result in radians, which is between - PI and PI (inclusive).

See also **acos()** and **atan()**.

base_convert

base_convert - - ¼ýÀÚ¸¦ Áø¹ý°£¿¡ °¯È¯ÇÑ´Ù.

Description

strin base_convert(string number, int frombase, int tobase);

Returns a string containing *number* represented in base *tobase*. The base in which *number* is given is specified in *frombase*. Both *frombase* and *tobase* have to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a - z, with a meaning 10, b meaning 11 and z meaning 36.
Example 1. base_convert()

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec

BinDec - - 2Áø¼ö¸¦ 10Áø¼ö·Î ¹Ù²Û´Ù.

Description

int bindec(string binary_string);

Returns the decimal equivalent of the binary number represented by the binary_string argument.

BinDec converts a binary number to a decimal number. The largest number that can be converted is 31 bits of 1's or 2147483647 in decimal.

See also the **decbin()** function.

Ceil

Ceil - - ¿Ã¸²À» ÇÑ °ª

Description

int ceil(float number);

Returns the next highest integer value from *number*. Using **ceil()** on integers is absolutely a waste of time.

NOTE: PHP/FI 2's **ceil()** returned a float. Use: `$new = (double)ceil($number);` to get the old behaviour.

See also **floor()** and **round()**.

Cos

Cos - - cosine

Description

float cos(float arg);

Returns the cosine of arg in radians.

See also **sin()** and **tan()**.

DecBin

DecBin - - 10Áø¼ö¸¦ 2Áø¼ö·Î ¹Ù²Û´Ù.

Description

string decbin(int number);

Returns a string containing a binary representation of the given number argument. The largest number that can be converted is 2147483647 in decimal resulting to a string of 31 1's.

See also the **bindec()** function.

DecHex

DecHex - - ½ÊÁø¼ö¸¦ 16Áø¼ö·Î

Description

string dechex(int number);

Returns a string containing a hexadecimal representation of the given number argument. The largest number that can be converted is 2147483647 in decimal resulting to "7fffffff".

See also the **hexdec()** function.

DecOct

DecOct - - ½ÊÁø¼ö¸¦ 8Áø¼ö·Î ¹Ù²Û´Ù

Description

string decoct(int number);

Returns a string containing an octal representation of the given number argument. The largest number that can be converted is 2147483647 in decimal resulting to "17777777777". See also **octdec()**.

Exp

Exp - - ÀÚ¿¬´ë¼ö eÀÇ n Á¦°ö°ª

Description

float exp(float arg);

Returns e raised to the power of *arg*.

See also **pow()**.

Floor

Floor - - ³»¸²Àº ÇÑ °ª

Description

int floor(float number);

Returns the next lowest integer value from *number*. Using **floor()** on integers is absolutely a waste of time.

NOTE: PHP/FI 2's **floor()** returned a float. Use: `$new = (double)floor($number);` to get the old behaviour.

See also **ceil()** and **round()**.

getrandmax

getrandmax - - °¡´ÉÇÑ ³-¼öÀÇ ÃÖ´ë°ª

Description

int getrandmax(void);

Returns the maximum value that can be returned by a call to **rand()**.

See also **rand()**, **srand()**, **mt - rand()** ,**mt_srand()** and **mt_getrandmax()**,

HexDec

HexDec - - 16Áø¼öº¸¦ 10Áø¼öº·Î

Description

int hexdec(string hex_string);

Returns the decimal equivalent of the hexadecimal number represented by the hex_string argument. HexDec converts a hexadecimal string to a decimal number. The largest number that can be converted is 7fffffff or 2147483647 in decimal.

See also the **dechex()** function.

Log

Log - - ÀÚ¿¬ ·Î±×

Description

float log(float arg);

Returns the natural logarithm of arg.

Log10

Log10 - - »ó¿ë ·Î±×

Description

float log10(float arg);

Returns the base - 10 logarithm of arg.

max

max - - °¡Àå Å« °ªÀ» Ã£´Â´Ù.

Description

mixed max(mixed arg1, mixed arg2, mixed argn);

max() returns the numerically highest of the parameter values.

If the first parameter is an array, **max()** returns the highest value in that array. If the first parameter is an integer, string or double, you need at least two parameters and **max()** returns the biggest of these values. You can compare an unlimited number of values.

If one or more of the values is a double, all the values will be treated as doubles, and a double is returned. If none of the values is a double, all of them will be treated as integers, and an integer is returned.

min

min - - °¡Àå ÀÛÀº °ªÀ» Ã£´Â´Ù.

Description

mixed min(mixed arg1, mixed arg2, mixed argn);

min() returns the numerically lowest of the parameter values.

If the first parameter is an array, **min()** returns the lowest value in that array. If the first parameter is an integer, string or double, you need at least two parameters and **min()** returns the lowest of these values. You can compare an unlimited number of values.

If one or more of the values is a double, all the values will be treated as doubles, and a double is returned. If none of the values is a double, all of them will be treated as integers, and an integer is returned.

mt_rand

mt_rand - - ´õ ÁÁÀº ³-¼ö °ªÀ» ¸¸µé¾î ³½´Ù.

Description

int mt_rand([int min], [int max]);

Many random number generators of older libcs have dubious or unknown characteristics and are slow. By default, PHP uses the libc random number generator with the **rand()** function. **mt_rand()** function is a drop-in replacement for this. It uses a random number generator with known characteristics, the Mersenne Twister, which will produce random numbers that should be suitable for cryptographic purposes and is four times faster than what the average libc provides. The Homepage of the Mersenne Twister can be found at http://www.math.keio.ac.jp/~matumoto/emt.html, and an optimized version of the MT source is available from http://www.scp.syr.edu/~marc/hawk/twister.html.

If called without the optional min,max arguments **mt_rand()** returns a pseudo-random value between 0 and RAND_MAX. If you want a random number between 5 and 15 (inclusive), for example, use mt_rand(5,15).

Remember to seed the random number generator before use with **mt_srand()**.

See also **mt_srand()**, **mt_getrandmax()**, **srand()**, **rand()** and **getrandmax()**.

mt_srand

mt_srand - - ´õ ÁÁÀº ³-¼ö ¹ß»ý±â¿¡ ÃÊ±â°ªÀ» Á¤ÇÑ´Ù.

Description

void mt_srand(int seed);

Seeds the random number generator with *seed*.

```
// seed with microseconds since last "whole" second
mt_srand((double)microtime()*1000000);
$randval = mt_rand();
```

See also **mt_rand()**, **mt_getrandmax()**, **srand()**, **rand()** and **getrandmax()**.

mt_getrandmax

mt_getrandmax - - °¡´ÉÑ °¡Àå Å« ³-¼ö°ª

Description

int mt_getrandmax(void);

Returns the maximum value that can be returned by a call to **mt_rand()**.

See also **mt_rand()**, **mt_srand()**, **rand()**, **srand()** and **getrandmax()**.

number_format

number_format - - ¼ýÀÚ¸¦ 1000´ÜÀ§¿Í ¼Ò¼öÁ¡¿¡ ,¿Í .µîÀ» Âï¾îÁØ´Ù.

format a number with grouped thousands

Description

`string number_format`(float number, int decimals, string dec_point, string thousands_sep);

number_format() returns a formatted version of *number*. This function accepts either one, two or four parameters (not three):

If only one parameter is given, *number* will be formatted without decimals, but with a comma (",") between every group of thousands.

If two parameters are given, *number* will be formatted with *decimals* decimals with a dot (".") in front, and a comma (",") between every group of thousands.

If all four parameters are given, *number* will be formatted with *decimals* decimals, *dec_point* instead of a dot (".") before the decimals and *thousands_sep* instead of a comma (",") between every group of thousands.

OctDec

OctDec - - 8Áø¼ö¸¦ 10Áø¼ö·Î

Description

int octdec(string octal_string);

Returns the decimal equivalent of the hexadecimal number represented by the hex_string argument. OctDec converts an octal string to a decimal number. The largest number that can be converted is 17777777777 or 2147483647 in decimal.

See also **decoct()**.

pi

pi - - ÆÄÀÌ °ª

Description

double pi(void);

Returns an approximation of pi.

pow

pow - - Áö¼ö Ç¥Çö½Ä(xÀÇ y½Â)

Description

float pow(float base, float exp);

Returns base raised to the power of exp.

See also **exp()**.

rand

rand - - ³¼öÀÇ ¹ß»ý

Description

`int rand`([int min], [int max]);

If called without the optional min,max arguments rand() returns a pseudo-random value between 0 and RAND_MAX. If you want a random number between 5 and 15 (inclusive), for example, use rand(5,15).

Remember to seed the random number generator before use with **srand()**.

See also **srand()**, **getrandmax()**, **mt - rand()** ,**mt_srand()** and **mt_getrandmax()**,

round

round - - ¹Ý¿Ã¸² ÇÑ °ª

Description

double round(double val);

Returns the rounded value of *val*.

```
$foo = round( 3.4 );   // $foo == 3.0
$foo = round( 3.5 );   // $foo == 4.0
$foo = round( 3.6 );   // $foo == 4.0
```

See also **ceil()** and **floor()**.

Sin

Sin - - sine

Description

float sin(float arg);

Returns the sine of arg in radians.

See also **cos()** and **tan()**.

Sqrt

Sqrt - - Á¦°ö±Ù

Description

float sqrt(float arg);

Returns the square root of arg.

srand

srand - - ³-¼ö ¹ß»ý½Ã »ç¿ëÇÏ´Â seed °ª

Description

void srand(int seed);

Seeds the random number generator with *seed*.

```
// seed with microseconds since last "whole" second
srand((double)microtime()*1000000);
$randval = rand();
```

See also **rand()**, **getrandmax()**, **mt - rand()** ,**mt_srand()** and **mt_getrandmax()**,

Tan

Tan - - tangent

Description

float tan(float arg);

Returns the tangent of arg in radians.

See also **sin()** and **cos()**.

XXVI. mcrypt functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº mcrypt¿Í °°ÀÌ µ¿ÀÛÇÏ±âÀ§ÇØ ¸¸µé¾ú´Ù.

ÀÌ°ÍÀº mcrypt ¶óÀ̰ê·¯¸®¿¡ ´ëÇÑ ÀÎÅÍÆäÀÌ½º°ú °°À° ±¤¹üÀ§ÇÑ block ¾Ë°í¸®ÁòÀ» Áö¿øÇÑ´Ù. : DES, TripleDES, Blowfish (default), 3 - WAY, SAFER - SK64, SAFER - SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes Ãß°¡·Î Áö¿øÇÏ´Â RC6¿Í IDEA´Â ¹«·á°¡ ¾Æ´Ï´Ù.

ÀÌ°ÍÀ» »ç¿ëÇÏ±â À§ÇØ¼´Â ¿ì¼± ftp://argeas.cs - net.gr/pub/unix/mcrypt/¿¡¼ libmcrypt - x.x.tar.gzÀ» ´Ù¿î¹ÞÀº ÈÄ¿¡ Æ÷ÇÔµÈ ¼³Ä¡ ¼³¸í¼ ´ë·Î ¼³Ä¡ÇÑ ÈÄ¿¡, PHP¸¦ - - with - mcrypt ¼³Á¤À» ÁÖ¾î ÀÌ È®ÀåÀ» È°¼º¾È-ÇÏ¿© ÄÄÆÄÀÏ ÇÏ¿©¾ß ÇÑ´Ù.

mcrypt´Â À§¿¡¼ ¾ð±ÞµÈ ¾ÏÈ£¹æ½Ä(cipher)À¸·Î ¾ÏÈ£È-ÇÏ°Å³ª ÇØµ¶ÇÒ ¼ö ÀÖ´Ù. **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ecb()** **mcrypt_ofb()**ÀÇ ³×°¡Áö Áß¿äÇÑ mcrypt ¸í·ÉÀº °¢°¢ MCRYPT_ENCRYPT¿Í MCRYPT_DECRYPT·Î ¸í¸íµÈ µÎ°¡Áö ¸ðµå·Î µ¿ÀÛÇÑ´Ù.

Example 1. Encrypt an input value with TripleDES in ECB mode

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

À§ÀÇ ¿¹Á¦¿¡¼ ¿©·¯°ÐÀº $encrypted_data¿¡ ¾ÏÈ£È-µÈ ¹®ÀÚ¿- µ¥ÀÌÅ¸¦ ¾ò°Ô µÈ´Ù.

mcrypt´Â CBC, OFB, CFB, ECBÀÇ 4°¡Áö ¾ÏÈ£ ¸ðµå·Î µ¿ÀÛÇÑ´Ù. ¿©±â¼´Â °³°³ÀÇ ¸ðµå¿¡ ´ëÇÑ °£·«ÇÑ ¾³¸í¸¸À» ÇÏ°Ú´Ù. °¸´Ù ÀÚ¼¼ÇÑ ³»¿ëÀº SchneierÀÇ Applied Cryptography (ISBN 0 - 471 - 11709 - 9)¸¦ °¸±â ¹Ù¶õ´Ù.

ECB(electronic codebook)´Â ´Ù¸¥ Å°¸¦ ¾ÏÈ£È-ÇÏ´Â µîÀÇ ·£´ý µ¥ÀÌÅÍ¿¡ ÀûÇÕÇÏ´Ù. µ¥ÀÌÅÍ° Â³°í ·£´ýÇÏ¹Ç·Î ECB´Â ´Ù¸¥ ¹æ½Ä¿¡ °ñÇØ °¸È¼öÁØÀÌ Á¶±Ý ³·Àº Æíİ´Ù.

CBC(cipher block chaining)´Â Æ¯È÷ ECB¿¡ °ñÇØ ³ôÀº °¸È ¼öÁØÀÌ ¿ä±¸Ç´Â ÆÄÀÏÀ» ¾ÏÈ£È-ÇÏ´Âµ¥ ÀûÇÕÇÏ´Ù.

CFB(cipher feedback)´Â °³°³ÀÇ ¹ÙÀÌÆ®°¡ ¾ÏÈ£È- µÇ¾î¾ß ÇÏ´Â byte streamÀ» ¾ÏÈ£È-ÇÏ´Âµ¥ °¡Àå ÁÁÀº ¸ðµåÀÌ´Ù.

OFB(output feedback)´Â CFB¿Í È£È¯µÇÁö¸¸, ¿¡·¯°¡ »ý±â¸é ¾ÈµÇ´Â ÀÀ¿ë ÇÁ·Î±×·¥¿¡ »ç¿ëÇÒ ¼ö ÀÖ´Ù.

PHP´Â ÇöÀç bit stream¿¡ ´ëÇØ¼´Â ¾ÏÈ£È-ÇÏ°Å³ª ÇØµ¶ÇÒ ¼ö ¾ø´Ù. ÇöÀçÀÇ PHP´Â ´ÜÁö ¹®ÀÚ¿-À» Áö¿øÇÒ »ÓÀÌ´Ù.

Áö¿øµÇ´Â ¾ÏÈ£¹æ½ÄÀÇ ÀüÃ¼ ¸®½º°Æ®´Â mcrypt.h ÆÄÀÌÀÇ ¸¶Áö¸·¿¡ ÀÖ´Ù. PHP¿¡¼ ¾ÏÈ£¹æ½Ä(cipher)À» »ç¿ëÇÏ´Â ÀÏ¹ÝÀûÀÎ ¹ýÄ¢À° "MCRYPT_¾ÏÈ£¹æ½ÄÀÌ¸§"ÀÇ ÇüÅÂÀÌ´Ù.

¾Æ·¡´Â ÇöÀç PHP3/mcrypt¿¡¼ Áö¿øÇÏ´Â ¾ÏÈ£¹æ½ÄÀÇ °£´ÜÇÑ ListÀÌ´Ù. mcrypt¿¡¼ Áö¿øÇÑ´Ù°í ÇÑ½ÃµÇ¾î ÀÖÀ¸³ª, ¿©±â¿¡ ³ª¿ÍÀÖÁö ¾ÊÀº °ÍÀº ÀÌ ¸Å´º¾ó ÀÌÈÄ¿¡ Ãß°¡·Î Áö¿øµÈ °ÍÀ̹Ç·Î ¾È½ÉÇÏ°í »ç¿ëÏ¿©µµ µÉ´Ù.

- 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

CFB¿Í OFB ¸ðµå¿¡¼´Â ¹Ýµå½Ã °¢°¢ÀÇ ¾ÏÈ£È- ÇÔ¼ö¿¡ ´ëÇØ initialization vector (IV)¸¦ ÁÖ¾î¾ß ÇÑ´Ù. CBC ¸ðµå¿¡¼´Â ¿É¼ÇÀ¸·Î ÁÙ ¼ö ÀÖ´Ù. IV´Â À¯ÀÏÇÏ¿©¾ß ÇÏ°í, ¾ÏÈ£/ÇØÁ¦½Ã¿¡ µ¿ÀÏÇÏ¿©¾ß ÇÑ´Ù. ¾ÏÈ£È-µÇ¾î ÀúÀåµÈ µ¥ÀÌÅÍ¸¦ °¡Áö°í, µ¥ÀÌÅÍ °¡ ÀúÅåµÈ°÷ÀÇ index¸¦ ¾î¶² ÇÔ¼ö¸¦ Âû¿ëÇÏ¿© ¿© ¾òÀº °ªÀ» »ç¿ëÇÒ ¼ö µµ ÀÖ´Ù. (¿¹, ÆÄÀÏÀÌ¸§ÀÇ MD5 Å°) ´Ù¸¥ ´ë¾ÈÀ¸·Î IV¸¦ ¾ÏÈ£È-µÈ µ¥ÀÌÅÍ¿Í ÇÔ²² º¸³»Â ¹æ¹ýÀÌ ÀÖ´Ù. (¿©±â¿¡ ´ëÇØ¼´Â SchneierÀÇ Applied CryptographyÀÇ chapter 9.3À» º¸¶ó.)

mcrypt_get_cipher_name

mcrypt_get_cipher_name - - Æ¯Á¤ ¾ÏÈ£¹æ½Ä(cipher)ÀÇ ÀÌ¸§À» ±¸ÇÑ´Ù.

Description

```
string mcrypt_get_cipher_name(int cipher);
```

mcrypt_get_cipher_name() is used to get the name of the specified cipher.

mcrypt_get_cipher_name() takes the cipher number as an argument and returns the name of the cipher.
Example 1. mcrypt_get_cipher_name example

```
<?php
$cipher = MCRYPT_TripleDES;
print mcrypt_get_cipher_name($cipher);
?>
```

The above example will produce:

```
TripleDES
```

mcrypt_get_block_size

mcrypt_get_block_size - - Æ¯Á¤ ¾ÏÈ£¹æ½Ä(cipher)ÀÇ °í·Ï Å©±â¸¦ ±¸ÇÑ´Ù.

Description

```
int mcrypt_get_block_size(int cipher);
```

mcrypt_get_block_size() is used to get the size of a block of the specified *cipher*.

mcrypt_get_block_size() takes one argument, the *cipher* and returns the size in bytes.

See also: **mcrypt_get_key_size()**

mcrypt_get_key_size

mcrypt_get_key_size - - Æ¯Á¤ ¾ÏÈ£¹æ½Ä(cipher)ÀÇ keyÀÇ ±æÀÌ¸¦ ±¸ÇÑ´Ù.

Description

```
int mcrypt_get_key_size(int cipher);
```

mcrypt_get_key_size() is used to get the size of a key of the specified *cipher*.

mcrypt_get_key_size() takes one argument, the *cipher* and returns the size in bytes.

See also: **mcrypt_get_block_size()**

mcrypt_create_iv

mcrypt_create_iv - - ·£´ý ¼Ò½º¿¡¼ initialization vector(IV)¸¦ ¸¸µç´Ù.

Description

`string mcrypt_create_iv`(int size, int source);

mcrypt_create_iv() is used to create an IV.

mcrypt_create_iv() takes two arguments, *size* determines the size of the IV, *source* specifies the source of the IV.

The source can be MCRYPT_RAND (system random number generator), MCRYPT_DEV_RANDOM (read data from /dev/random) and MCRYPT_DEV_URANDOM (read data from /dev/urandom). If you use MCRYPT_RAND, make sure to call srand() before to initialize the random number generator.
Example 1. mcrypt_create_iv example

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size($cipher);
$iv = mcrypt_create_iv($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_cbc

mcrypt_cbc - - CBC ¸ðµå·Î µ¥ÀÌÅÍ¸¦ ¾ÏÈ£È-ÇÏ°Å³ª ÇØÁ¦ ÇÑ´Ù.

Description

`int mcrypt_cbc`(int cipher, string key, string data, int mode, string [iv]);

mcrypt_cbc() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CBC cipher mode and returns the resulting string.

cipher is one of the MCRYPT_ciphername constants.

key is the key supplied to the algorithm. It must be kept secret.

data is the data which shall be encrypted/decrypted.

mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

iv is the optional initialization vector.

See also: **mcrypt_cfb()**, **mcrypt_ecb()**, **mcrypt_ofb()**

mcrypt_cfb

mcrypt_cfb - - CFB ¸ðµå·Î µ¥ÀÌÅÍ¸¦ ¾ÏÈ£È-ÇÏ°Å³ª ÇØÁ¦ ÇÑ´Ù.

Description

`int mcrypt_cfb`(int cipher, string key, string data, int mode, string iv);

mcrypt_cfb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CFB cipher mode and returns the resulting string.

cipher is one of the MCRYPT_ciphername constants.

key is the key supplied to the algorithm. It must be kept secret.

data is the data which shall be encrypted/decrypted.

mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

iv is the initialization vector.

See also: **mcrypt_cbc()**, **mcrypt_ecb()**, **mcrypt_ofb()**

mcrypt_ecb

mcrypt_ecb - - ECB ¸ðµå·Î µ¥ÀÌÅÍ¸¦ ¾ÏÈ£È-ÇÏ°Å³ª ÇØÁ¦ ÇÑ´Ù.

Description

int mcrypt_ecb(int cipher, string key, string data, int mode);

mcrypt_ecb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in ECB cipher mode and returns the resulting string.

cipher is one of the MCRYPT_ciphername constants.

key is the key supplied to the algorithm. It must be kept secret.

data is the data which shall be encrypted/decrypted.

mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

See also: **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ofb()**

mcrypt_ofb

mcrypt_ofb - - OFB ¸ðµå·Î µ¥ÀÌÅÍ¸¦ ¾ÏÈ£È-ÇÏ°Å³ª ÇØÁ¦ ÇÑ´Ù.

Description

int mcrypt_ofb(int cipher, string key, string data, int mode, string iv);

mcrypt_ofb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in OFB cipher mode and returns the resulting string.

cipher is one of the MCRYPT_ciphername constants.

key is the key supplied to the algorithm. It must be kept secret.

data is the data which shall be encrypted/decrypted.

mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

iv is the initialization vector.

See also: **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ecb()**

XXVII. Miscellaneous Functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº ±âÁ¸ ¹üÁÖ¿¡ Æ÷ÇÔ½ÅÅ°±â °ï¶õÇÑ °ÍµéÀÌ´Ù.

connection_aborted

connection_aborted - - Å¬¶óÀÌ¾ðÆ®°¡ Á¢¼ÓÀ» ²÷¾ú´Ù¸é true¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int connection_aborted(void);

Returns true if client disconnected. See the Connection Handling description in the Feature chapter for a complete explanation.

connection_status

connection_status - - ¿¬°á »óÅÂ¸¦ bit´ÜÀ§ÀÇ °ªÀ¸·Î ¹ÝÈ¯ÇÑ´Ù.

Description

int connection_status(void);

Returns the connection status bitfield. See the Connection Handling description in the Feature chapter for a complete explanation.

connection_timeout

connection_timeout - - ½º¼¹ö¸¦ ¼³Á¤µÈ ½Ã°£À» ÀÊ°úÇÏ¿© timed outµÇ¾úÀ¸¸é true¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int connection_timeout(void);

Returns true if script timed out. See the Connection Handling description in the Feature chapter for a complete explanation.

eval

eval - - PHP ÄÚµå¸¦ °¡Áø ¹®ÀÚ¿À» ½ÇÇàÇÑ´Ù.

Description

void eval(string code_str);

eval() evaluates the string given in *code_str* as PHP code. Among other things, this can be useful for storing code in a database text field for later execution.

There are some factors to keep in mind when using **eval()**. Remember that the string passed must be valid PHP code, including things like terminating statements with a semicolon so the parser doesn't die on the line after the **eval()**, and properly escaping things in *code_str*.

Also remember that variables given values under **eval()** will retain these values in the main script afterwards.
Example 1. eval() example - simple text merge

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
?>
```

The above example will show:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

extract

extract - - ÇÑ ¹è¿¿¡¼ symbol table·Î °¡¼öµéÀ» ÀÐ¾î µéÀÎ´Ù.

Description

void extract(array var_array, int extract_type, string [var_array]);

This function is used to import variables from an array into the current symbol table. It takes associative array *var_array* and treats keys as variable names and values as variable values. For each key/value pair it will create a variable in the current symbol table, subject to *extract_type* and *prefix* parameters.

extract() checks for colissions with existing variables. The way collisions are treated is determined by *extract_type*. It can be one of the following values:

EXTR_OVERWRITE
    If there is a collision, overwrite the existing variable.
EXTR_SKIP

    If there is a collision, don't overwrite the existing variable.

    EXTR_PREFIX_SAME
        If there is a collision, prefix the new variable with *prefix*.
    EXTR_PREFIX_ALL
        Prefix all variables with *prefix*.

Note that *prefix* is only required if *extract_type* is EXTR_PREFIX_SAME or EXTR_PREFIX_ALL.

extract() checks each key to see if it constitues a valid variable name, and if it does only then does it proceed to import it.

A possible use for extract is to import into symbol table variables contained in an associative array returned by **wddx_deserialize()**.
Example 1. extract example

```
<?
/* Suppose that $var_array is an array returned from
   wddx_deserialize */
$size = "large";
$var_array = array("color" => "blue",
                   "size"  => "medium",
                   "shape" => "sphere");
extract($var_array, EXTR_PREFIX_SAME, "wddx");
print "$color, $size, $shape, $wddx_size\n";
?>
```

The above example will produce:

```
blue, large, sphere, medium
```

The $size wasn't overwritten, becaus we specified EXTR_PREFIX_SAME, which resulted in $wddx_size being created. If EXTR_SKIP was specified, then $wddx_size wouldn't even have been created. EXTR_OVERWRITE would have cause $size to have value "medium", and EXTR_PREFIX_ALL would result in new variables being named $wddx_color, $wddx_size, and $wddx_shape.

die

die - - ¸Þ½ÃÁö¸¦ Ãâ·ÂÇÏ°í ÇöÀç ½ºÅ©¸³Æ®ÀÇ ½ÇÇàÀ» Áß´ÜÇÑ´Ù.

Description

```
void die(string message);
```

This language construct outputs a message and terminates parsing of the script. It does not return.
Example 1. die example

```
<?php
$filename = '/path/to/data-file';
$file = fopen($filename, 'r')
  or die "unable to open file ($filename)";
?>
```

exit

exit - - ÇöÀç ½ºÅ©¸³Æ®ÀÇ ½ÇÇàÀ» Áß´ÜÇÑ´Ù.

Description

```
void exit(void);
```

This language construct terminates parsing of the script. It does not return.

function_exists

function_exists - - ÁÖ¾îÁø ÀÌ¸§ÀÇ ÇÔ¼ö°¡ Á¤ÀÇµÇ¾î ÀÖ´Ù¸é true¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int function_exists(string function_name);

Checks the list of defined functions for *function_name*. Returns true if the given function name was found, false otherwise.

ignore_user_abort

ignore_user_abort - - Å¬¶óÀÌ¾ðÆ®°¡ Á¢¼ÓÀ» ²÷¾úÀ» ¶§ ½ºÅ©¸³Æ®ÀÇ ¼öÇàÀ» ÁßÁöÇÏ´Â°¡¸¦ Á¤ÇÑ´Ù.

Description

int ignore_user_abort(int [setting]);

This function sets whether a client disconnect should cause a script to be aborted. It will return the previous setting and can be called without an argument to not change the current setting and only return the current setting. See the Connection Handling section in the Features chapter for a complete description of connection handling in PHP.

iptcparse

iptcparse - - binary IPTC http://www.xe.net/iptc/ blockÀ» ÇÏ³ªÀÇ ÀÃÀ¸·Î parseÇÑ´Ù.

Description

array iptcparse(string iptcblock);

This function parses a binary IPTC block into its single tags. It returns an array using the tagmarker as an index and the value as the value. See **GetImageSize()** for a sample.

leak

leak - - ¸Þ¸ð¸®¸¦ ´©¼³(Leak) ÇÑ´Ù.

Description

void leak(int bytes);

Leak() leaks the specified amount of memory.

This is useful when debugging the memory manager, which automatically cleans up "leaked" memory when each request is completed.

pack

pack - - data¸¦ binary ¹®ÀÚ¿·Î packÇÑ´Ù.

Description

string pack(string format, mixed [args]...);

Pack given arguments into binary string according to *format*. Returns binary string containing data.

The idea to this function was taken from Perl and all formatting codes work the same as there. The format string consists of format codes followed by an optional repeater argument. The repeater argument can be either an integer value or * for repeating to the end of the input data. For a, A, h, H the repeat count specifies how many characters of one data argument are taken, for @ it is the absolute position where to put the next data, for everything else the repeat count specifies how many data arguments are consumed and packed into the resulting binary string. Currently implemented are

a NUL - padded string
A SPACE - padded string
h Hex string, low nibble first
H Hex string, high nibble first
c signed char
C unsigned char

s signed short (always 16 bit, machine byte order)
S unsigned short (always 16 bit, machine byte order)
n unsigned short (always 16 bit, big endian byte order)
v unsigned short (always 16 bit, little endian byte order)
i signed integer (machine dependant size and byte order)
I unsigned integer (machine dependant size and byte order)
l signed long (always 32 bit, machine byte order)
L unsigned long (always 32 bit, machine byte order)
N unsigned long (always 32 bit, big endian byte order)
V unsigned long (always 32 bit, little endian byte order)
f float (machine dependent size and representation)
d double (machine dependent size and representation)
x NUL byte
X Back up one byte
@ NUL - fill to absolute position

Example 1. pack format string

```
$binarydata = pack("nvc*", 0x1234, 0x5678, 65, 66);
```

The resulting binary string will be 6 bytes long and contain the byte sequence 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Note that the distinction between signed and unsigned values only affects the function **unpack()**, where as function **pack()** gives the same result for signed and unsigned format codes.

Also note that PHP internally stores integral values as signed values of a machine dependant size. If you give it an unsigned integral value too large to be stored that way it is converted to a double which often yields an undesired result.

register_shutdown_function

register_shutdown_function - - ½º©¸³Æ®°¡ Á¾·áµÉ ¶§ ½ÇÇàµÉ ÇÔ¼ö¸¦ ¼³Á¤ÇÑ´Ù.

Description

```
int register_shutdown_function(string func);
```

Registers the function named by *func* to be executed when script processing is complete.

serialize

serialize - - ¼îÈ² ÇüÅÂÀÇ °ªÀÌ¶óµµ ÀúÀå °¡´ÉÇÑ ÇüÅÂÀÇ ¹®ÀÚ¿·Î ¸¸µé¾î ÁØ´Ù.

Description

```
string serialize(mixed value);
```

serialize() returns a string containing a byte - stream representation of *value* that can be stored anywhere.

This is useful for storing or passing PHP values around without losing their type and structure.

To make the serialized string into a PHP value again, use **unserialize()**. **serialize()** handles the types integer, double, string, array (multidimensional) and object (object properties will be serialized, but methods are lost).

Example 1. serialize example

```
// $session_data contains a multi-dimensional array with session
// information for the current user.  We use serialize() to store
// it in a database at the end of the request.
$conn = odbc_connect("webdb", "php", "chicken");
$stmt = odbc_prepare($conn,
                     "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array(serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
                         "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
        /* Something went wrong.  Bitch, whine and moan. */
    }
}
```

sleep

sleep - - ½ÇÇàÀ» Àá½Ã Áö¿¬½ÃÅ²´Ù.

Description

void sleep(int seconds);

The sleep function delays program execution for the given number of *seconds*.

See also **usleep()**.

unpack

unpack - - binary ¹®ÀÚ¿-À» data·Î unpackÇÑ´Ù.

Description

array unpack(string format, string data);

Unpack from binary string into array according to *format*. Returns array containing unpacked elements of binary string.

Unpack works slightly different from Perl as the unpacked data is stored in an associative array. To accomplish this you have to name the different format codes and separate them by a slash /.
Example 1. unpack format string

```
$array = unpack("c2chars/nint", $binarydata);
```

The resulting array will contain the entries "chars1", "chars2" and "int".

For an explanation of the format codes see also: **pack()**

Note that PHP internally stores integral values as signed. If you unpack a large unsigned long and it is of the same size as PHP internally stored values the result will be a negative number even though unsigned unpacking was specified.

unserialize

unserialize - - ÀúÀå°¡´ÉÑ ÇüÅÂ·Î Ç¥ÇöµÈ ¹®ÀÚ¿-À» PHP °ªÀ¸·Î ¸¸µé¾îÁØ´Ù.

creates a PHP value from a stored representation

Description

mixed unserialize(string str);

unserialize() takes a single serialized variable (see **serialize()**) and converts it back into a PHP value. The converted value is returned, and can be an integer, double, string, array or object. If an object was serialized, its methods are not preserved in the returned value.
Example 1. unserialize example

```
// Here, we use unserialize() to load session data from a database
// into $session_data.  This example complements the one described
// with serialize().
$conn = odbc_connect("webdb", "php", "chicken");
$stmt = odbc_prepare($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array($PHP_AUTH_USER);
if (!odbc_execute($stmt, &$sqldata) || !odbc_fetch_into($stmt, &$tmp)) {
    // if the execute or fetch fails, initialize to empty array
    $session_data = array();
} else {
    // we should now have the serialized data in $tmp[0].
    $session_data = unserialize($tmp[0]);
    if (!is_array($session_data)) {
        // something went wrong, initialize to empty array
        $session_data = array();
    }
}
```

uniqid

uniqid - - À¯ÀÏÇÑ id¸¦ »ý¼ºÇÑ´Ù.

Description

int uniqid(string prefix);

uniqid() returns a prefixed unique identifier based on current time in microseconds. The prefix can be useful for instance if you generate identifiers simultaneously on several hosts that might happen to generate the identifier at the same microsecond. The prefix can be up to 114 characters long.

If you need a unique identifier or token and you intend to give out that token to the user via the network (i.e. session cookies), it is recommended that you use something along the lines of

```
$token = md5(uniqid("")); // no random portion
$better_token = md5(uniqid(random())); // better, difficult to guess
```

This will create a 32 character identifier (a 128 bit hex number) that is extremely difficult to predict.

usleep

usleep - - ½ÇÇàÀ» ¹é¸¸ºÐÀÇ ÀÏÃÂÊ ´ÜÀ§·Î Áö¿¬½ÃÅ²´Ù.

Description

void usleep(int micro_seconds);

The sleep function delays program execution for the given number of *micro_seconds*.

See also **sleep()**.

XXVIII. mSQL Functions

Table of Contents

msql

msql - - mSQL ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

int msql(string database, string query, int link_identifier);

Returns a positive mSQL result identifier to the query result, or false on error.

msql() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the mSQL server and if no such link is found it'll try to create one as if **msql_connect()** was called with no arguments (see **msql_connect()**).

msql_affected_rows

msql_affected_rows - - ÃÖ±Ù ÁúÀÇ¿¡ ¿µÇâÀ» ¹Þ´Â rowÀÇ °³¼ö¸¦ ±¸Ñ´Ù.

Description

int msql_affected_rows(int query_identifier);

Returns number of affected ("touched") rows by a specific query (i.e. the number of rows returned by a SELECT, the number of rows modified by an update, or the number of rows removed by a delete).

See also: **msql_query()**

msql_close

msql_close - - mSQL connectionÀ» ´Ý´Â´Ù.

Description

int msql_close(int link_identifier);

Returns true on success, false on error.

msql_close() closes the link to a mSQL database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

msql_close() will not close persistent links generated by **msql_pconnect()**.

See also: **msql_connect()** and **msql_pconnect()**.

msql_connect

msql_connect - - mSQL connectionÀ» ¿¬´Ù.

Description

int msql_connect(string hostname);

Returns a positive mSQL link identifier on success, or false on error.

msql_connect() establishes a connection to a mSQL server. The hostname argument is optional, and if it's missing, localhost is assumed.

In case a second call is made to msql_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **msql_close()**.

See also **msql_pconnect()**, **msql_close()**.

msql_create_db

msql_create_db - - mSQL database¸¦ ¸¸µç´Ù.

Description

int msql_create_db(string database name, int [link_identifier]);

msql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: **msql_drop_db()**.

msql_createdb

msql_createdb - - mSQL database¸¦ ¸¸µç´Ù.

Description

int msql_createdb(string database name, int [link_identifier]);

Identical to **msql_create_db()**.

msql_data_seek

msql_data_seek - - ³»°ÎÀûÀÎ row pointer¸¦ ÀÌµ¿ÇÑ´Ù.

Description

int msql_data_seek(int result_identifier, int row_number);

Returns true on success, false on failure.

msql_data_seek() moves the internal row pointer of the mSQL result associated with the specified result identifier to pointer to the specifyed row number. The next call to **msql_fetch_row()** would return that row.

See also: **msql_fetch_row()**.

msql_dbname

msql_dbname - - ÇöÀç mSQL databaseÀÇ À̸§À» ±¸ÇÑ´Ù.

Description

string msql_dbname(string result, int i);

msql_dbname() returns the database name stored in position *i* of the result pointer returned from the **msql_listdbs()** function. The **msql_numrows()** function can be used to determine how many database names are available.

msql_drop_db

msql_drop_db - - mSQL database¸¦ ¹ö¸°´Ù.(drop = delete)

Description

int msql_drop_db(string database_name, int link_identifier);

Returns true on success, false on failure.

msql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: **msql_create_db()**.

msql_dropdb

msql_dropdb - - mSQL database¸¦ ¹ö¸°´Ù.(drop = delete)

Description

See **msql_drop_db()**.

msql_error

msql_error - - ¸¶Áö¸· mSQL È£ÃâÀÇ ¿¡·¯ ¸Þ¼¼Áö¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

string msql_error( );

Errors coming back from the mSQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

msql_fetch_array

msql_fetch_array - - row¸¦ ¹è¿·Î °¡Á®¿Â´Ù.

Description

int msql_fetch_array(int result);

Returns an array that corresponds to the fetched row, or false if there are no more rows.

msql_fetch_array() is an extended version of **msql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

Be careful if you are retrieving results from a query that may return a record that contains only one field that has a value of 0 (or an empty string, or NULL).

An important thing to note is that using msql_fetch_array() is NOT significantly slower than using **msql_fetch_row()**, while it provides a significant added value.

For further details, also see **msql_fetch_row()**

msql_fetch_field

msql_fetch_field - - ÇÊµå Á¤º¸¸¦ ±¸Ñ´Ù.

Description

object msql_fetch_field(int result, int field_offset);

Returns an object containing field information

msql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by msql_fetch_field() is retreived.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- not_null - 1 if the column cannot be null
- primary_key - 1 if the column is a primary key
- unique - 1 if the column is a unique key
- type - the type of the column

See also **msql_field_seek()**.

msql_fetch_object

msql_fetch_object - - row¸¦ °´Ã¼(Object)·Î °¡Á®¿Â´Ù.

Description

int msql_fetch_object(int result);

Returns an object with properties that correspond to the fetched row, or false if there are no more rows.

msql_fetch_object() is similar to **msql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **msql_fetch_array()**, and almost as quick as **msql_fetch_row()** (the difference is insignificant).

See also: **msql_fetch_array()** and **msql_fetch_row()**.

msql_fetch_row

msql_fetch_row - - row¸¦ ¹è¿-(enumerated array)·Î °¡Á®¿Â´Ù.

Description

array msql_fetch_row(int result);

Returns an array that corresponds to the fetched row, or false if there are no more rows.

msql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to msql_fetch_row() would return the next row in the result set, or false if there are no more rows.

See also: **msql_fetch_array()**, **msql_fetch_object()**, **msql_data_seek()**, and **msql_result()**.

msql_fieldname

msql_fieldname - - ÇÊµå ÀÌ¸§À» ±¸Ñ´Ù.

Description

string msql_fieldname(int result, int field);

msql_fieldname() returns the name of the specified field. *result* is the result identifier, and *field* is the field index. `msql_fieldname($result, 2);` will return the name of the second field in the result associated with the result identifier.

msql_field_seek

msql_field_seek - - ÇÊµåÀÇ offsetÀ» ¼³Á¤ÇÑ´Ù.

Description

int msql_field_seek(int result, int field_offset);

Seeks to the specified field offset. If the next call to **msql_fetch_field()** won't include a field offset, this field would be returned.

See also: **msql_fetch_field()**.

msql_fieldtable

msql_fieldtable - - ÇØ´ç ÇÊµå¸¦ °¡Á® ¿Â Table ÀÌ¸§À» ±¸Ñ´Ù.

Description

int msql_fieldtable(int result, int field);

Returns the name of the table *field* was fetched from.

msql_fieldtype

msql_fieldtype - - ÇÊµå typeÀ» ±¸ÇÑ´Ù.

Description

string msql_fieldtype(string result, int i);

msql_fieldtype() is similar to the **msql_fieldname()** function. The arguments are identical, but the field type is returned. This will be one of "int", "string" or "real".

msql_fieldflags

msql_fieldflags - - ÇÊµå flag¸¦ ±¸ÇÑ´Ù.

Description

string msql_fieldflags(string result, int i);

msql_fieldflags() returns the field flags of the specified field. Currently this is either, "not null", "primary key", a combination of the two or "" (an empty string).

msql_fieldlen

msql_fieldlen - - ÇÊµå ±æÀÌ¸¦ ±¸ÇÑ´Ù.

Description

int msql_fieldlen(string result, int i);

msql_fieldlen() returns the length of the specified field.

msql_free_result

msql_free_result - - result memory¸¦ Ç®¾îÁØ´Ù.

Description

int msql_free_result(int result);

msql_free_result() frees the memory associated with *result*. When PHP completes a request, this memory is freed automatically, so you only need to call this function when you want to make sure you don't use too much memory while the script is running.

msql_freeresult

msql_freeresult - - result memory¸¦ Ç®¾îÁØ´Ù.

Description

See **msql_free_result()**

msql_list_fields

msql_list_fields - - result fieldµéÀ» ³ª¿-ÇÑ´Ù.

Description

int msql_list_fields(string database, string tablename);

msql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **msql_fieldflags()**, **msql_fieldlen()**, **msql_fieldname()**, and **msql_fieldtype()**. A result identifier is a positive integer. The function returns `-1` if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@msql_list_fields()` then this error string will also be printed out.

See also **msql_error()**.

msql_listfields

msql_listfields - - result fieldµéÀ» ³ª¿-ÇÑ´Ù.

Description

See **msql_list_fields()**.

msql_list_dbs

msql_list_dbs - - serverÀÇ mSQL databaseµéÀ» ³ª¿-ÇÑ´Ù.

Description

int msql_list_dbs(void);

msql_list_dbs() will return a result pointer containing the databases available from the current msql daemon. Use the **msql_dbname()** function to traverse this result pointer.

msql_listdbs

msql_listdbs - - serverÀÇ mSQL databaseµéÀ» ³ª¿-ÇÑ´Ù.

Description

See **msql_list_dbs()**.

msql_list_tables

msql_list_tables - - mSQL databaseÀÇ tableµéÀ» ³ª¿-ÇÑ´Ù.

Description

int msql_list_tables(string database);

msql_list_tables() takes a database name and result pointer much like the **msql()** function. The **msql_tablename()** function should be used to extract the actual table names from the result pointer.

msql_listtables

msql_listtables - - mSQL databaseÀÇ tableµéÀ» ³ª¿-ÇÑ´Ù.

Description

See **msql_list_tables()**.

msql_num_fields

msql_num_fields - - resultÀÇ field °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int msql_num_fields(int result);

msql_num_fields() returns the number of fields in a result set.

See also: **msql()**, **msql_query()**, **msql_fetch_field()**, and **msql_num_rows()**.

msql_num_rows

msql_num_rows - - resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int msql_num_rows(string result);

msql_num_rows() returns the number of rows in a result set.

See also: **msql()**, **msql_query()**, and **msql_fetch_row()**.

msql_numfields

msql_numfields - - resultÀÇ field °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int msql_numfields(void);

Identical to **msql_num_fields()**.

msql_numrows

msql_numrows - - resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int msql_numrows(void);

Identical to **msql_num_rows()**.

msql_pconnect

msql_pconnect - - ¿µ±¸ÀûÀÎ mSQL Á¢¼ÓÀ» ¿¬´Ù.

Description

int msql_pconnect(string hostname);

Returns a positive mSQL persistent link identifier on success, or false on error.

msql_pconnect() acts very much like **msql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**msql_close()** will not close links established by msql_pconnect()).

This type of links is therefore called 'persistent'.

msql_query

msql_query - - mSQL ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

int msql_query(string query, int link_identifier);

msql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **msql_connect()** was called, and use it.

Returns a positive mSQL result identifier on success, or false on error.

See also: **msql()**, **msql_select_db()**, and **msql_connect()**.

msql_regcase

msql_regcase - - Á¤±Ô Ç¥Çö½Ä(regular expression)À» ´ë,¼Ò¹®ÀÚ¸¦ °¡¸®¾Æ¾ö ¾Êµµ·Ï ÇÑ´Ù.

Description

See **sql_regcase()**.

msql_result

msql_result - - result data¸¦ ±¸ÇÑ´Ù.

Description

int msql_result(int result, int i, mixed field);

Returns the contents of the cell at the row and offset in the specified mSQL result set.

msql_result() returns the contents of one cell from a mSQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than msql_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high - performance alternatives: **msql_fetch_row()**, **msql_fetch_array()**, and **msql_fetch_object()**.

msql_select_db

msql_select_db - - mSQL database¸¦ ¼±ÅÃÇÑ´Ù.

Description

int msql_select_db(string database_name, int link_identifier);

Returns true on success, false on error.

msql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if msql_connect() was called, and use it.

Every subsequent call to **msql_query()** will be made on the active database.

See also: **msql_connect()**, **msql_pconnect()**, and **msql_query()**.

msql_selectdb

msql_selectdb - - mSQL database¸¦ ¼±ÅÃÇÑ´Ù.

Description

See **msql_select_db()**.

msql_tablename

msql_tablename - - ÇØ´ç ÇÊµå¸¦ °¡Á® ¿Â Table ÀÌ¸§À» ±¸ÇÑ´Ù.

Description

string msql_tablename(int result, int field);

msql_tablename() takes a result pointer returned by the **msql_list_tables()** function as well as an integer index and returns the name of a table. The **msql_numrows()** function may be used to determine the number of tables in the result pointer.

Example 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables("wisconsin");
$i = 0;
while ($i < msql_numrows($result)) {
    $tb_names[$i] = msql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XXIX. MS SQL Server Functions

Table of Contents

mssql_affected_rows

mssql_affected_rows - - ÃÖ±Ù ÁúÀÇ¿¡ ¿µÇâÀ» ¹Þ´Â rowÀÇ °³¼ö¸¦ ±¸Ñ´Ù.

Description

int mssql_affected_rows(int [link_identifier]);

Returns: The number of affected rows by the last query.

mssql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use **mssql_num_rows()**.

mssql_close

mssql_close - - MS SQL Server connectionÀ» ´Ý´Â´Ù.

Description

int mssql_close(int link_identifier);

Returns: true on success, false on error

mssql_close() closes the link to a MS SQL Server database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

mssql_close() will not close persistent links generated by mssql_pconnect().

See also: **mssql_connect()**, **mssql_pconnect()**.

mssql_connect

mssql_connect - - MS SQL server connectionÀ» ¿¬´Ù.

Description

int mssql_connect(string servername, string username, string password);

Returns: A positive MS SQL link identifier on success, or false on error.

mssql_connect() establishes a connection to a MS SQL server. The servername argument has to be a valid

servername that is defined in the 'interfaces' file.

In case a second call is made to mssql_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **mssql_close()**.

See also **mssql_pconnect()**, **mssql_close()**.

mssql_data_seek

mssql_data_seek - - internal row pointer¸¦ ¿Å±ä´Ù.

Description

`int mssql_data_seek`(int result_identifier, int row_number);

Returns: true on success, false on failure

mssql_data_seek() moves the internal row pointer of the MS SQL result associated with the specified result identifier to pointer to the specifyed row number. The next call to **mssql_fetch_row()** would return that row.

See also: **mssql_data_seek()**.

mssql_fetch_array

mssql_fetch_array - - row¸¦ ¹è¿-·Î °¡Á®¿Â´Ù.

Description

`int mssql_fetch_array`(int result);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

mssql_fetch_array() is an extended version of **mssql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using mssql_fetch_array() is NOT significantly slower than using mssql_fetch_row(), while it provides a significant added value.

For further details, also see **mssql_fetch_row()**

mssql_fetch_field

mssql_fetch_field - - ÇÊµåÀÇ Á¤º¸¸¦ ±¸Ñ´Ù.

Description

`object mssql_fetch_field`(int result, int field_offset);

Returns an object containing field information.

mssql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by mssql_fetch_field() is retreived.

The properties of the object are:

name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.

column_source - the table from which the column was taken

max_length - maximum length of the column

numeric - 1 if the column is numeric

See also **mssql_field_seek()**

mssql_fetch_object

mssql_fetch_object - - row¸¦ object·Î °¡Á®¿Â´Ù.

Description

int mssql_fetch_object(int result);

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

mssql_fetch_object() is similar to **mssql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed - wise, the function is identical to **mssql_fetch_array()**, and almost as quick as **mssql_fetch_row()** (the difference is insignificant).

See also: **mssql_fetch - array()** and **mssql_fetch - row()**.

mssql_fetch_row

mssql_fetch_row - - row¸¦ ¹è¿-(enumerated array)·Î °¡Á®¿Â´Ù.

Description

array mssql_fetch_row(int result);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

mssql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to mssql_fetch_rows() would return the next row in the result set, or false if there are no more rows.

See also: **mssql_fetch_array()**, **mssql_fetch_object()**, **mssql_data_seek()**, **mssql_fetch_lengths()**, and **mssql_result()**.

mssql_field_seek

mssql_field_seek - - ÇÊµåÀÇ offsetÀ» ¼³Á¤ÇÑ´Ù.

Description

int mssql_field_seek(int result, int field_offset);

Seeks to the specified field offset. If the next call to **mssql_fetch_field()** won't include a field offset, this field would be returned.

See also: **mssql_fetch_field()**.

mssql_free_result

mssql_free_result - - result memory¸¦ Ç®¾îÁØ´Ù.

Description

int mssql_free_result(int result);

mssql_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script, you may call **mssql_free_result()** with the result identifier as an argument and the associated result memory will be freed.

mssql_num_fields

mssql_num_fields - - resultÀÇ field °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int mssql_num_fields(int result);

mssql_num_fields() returns the number of fields in a result set.

See also: **mssql_db_query()**, **mssql_query()**, **mssql_fetch_field()**, **mssql_num_rows()**.

mssql_num_rows

mssql_num_rows - - resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int mssql_num_rows(string result);

mssql_num_rows() returns the number of rows in a result set.

See also: **mssql_db_query()**, **mssql_query()** and, **mssql_fetch_row()**.

mssql_pconnect

mssql_pconnect - - ¿µ±¸ÀûÀÎ(persistent) MS SQL connectionÀ» ¿¬°áÇÑ´Ù.

Description

int mssql_pconnect(string servername, string username, string password);

Returns: A positive MS SQL persistent link identifier on success, or false on error

mssql_pconnect() acts very much like **mssql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**mssql_close()** will not close links established by mssql_pconnect()).

This type of links is therefore called 'persistent'.

mssql_query

mssql_query - - MS SQL ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

int mssql_query(string query, int link_identifier);

Returns: A positive MS SQL result identifier on success, or false on error.

mssql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **mssql_connect()** was called, and use it.

See also: **mssql_db_query()**, **mssql_select_db()**, and **mssql_connect()**.

mssql_result

mssql_result - - result data¸¦ ±¸ÇÑ´Ù.

Description

int mssql_result(int result, int i, mixed field);

Returns: The contents of the cell at the row and offset in the specified MS SQL result set.

mssql_result() returns the contents of one cell from a MS SQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than mssql_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: **mssql_fetch_row()**, **mssql_fetch_array()**, and **mssql_fetch_object()**.

mssql_select_db

mssql_select_db -- »ç¿ëÇÒ MS SQL database¸¦ ¼±ÅÃÇÑ´Ù.

Description

`int mssql_select_db`(string database_name, int link_identifier);

Returns: true on success, false on error

mssql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mssql_connect()** was called, and use it.

Every subsequent call to **mssql_query()** will be made on the active database.

See also: **mssql_connect()**, **mssql_pconnect()**, and **mssql_query()**

XXX. MySQL Functions

Table of Contents

These functions allow you to access MySQL database servers.

More information about MySQL can be found at http://www.mysql.com/.

(¿ªÀÚÁÖ : **Solaris MySQL with pthreads note:**
¸¸¾à ¿©·¯ºÐÀÌ MySQLÀÌ threaded client library(default´Â ¾Æ´Ï´Ù)¸¦ »ç¿ëÇÏ°í, PHP¸¦ MySQLÁö¿øÀ¸·Î ÄÄÆÄÀÏÇÑ´Ù¸é ¾Æ¸¶ ¿©·¯ºÐÀº ´ÙÀ½°ú °°Àº ¿¡·¯¸¦ ¸¸³ª°Ô µÈ´Ù....

```
Undefined first referenced
symbol in file
pthread_attr_setschedparam /opt/GNUmysql/lib/mysql/libmysqlclient.a(my_pthread.o)
pthread_setschedparam /opt/GNUmysql/lib/mysql/libmysqlclient.a(my_pthread.o)
```

ÀÌ¶§´Â MakefileÀ» ¼ÕÀ¸·Î °íÃÄÁÖ¾î¾ß ÇÑ´Ù. "LIBS"·Î ½ÃÀÛµÇ´Â ÁÙÀ» Ã£¾Æ "-lpthreads",¦ ÁÙÀÇ ¸¶Áö¸·¿¡ »ðÀÔÇÑ´Ù. ±×¸®°í "make"ÇÏ¸é µÉ °ÍÀÌ´Ù.)

---

mysql_affected_rows

mysql_affected_rows -- ÃÖ±Ù ÁúÀÇ¿¡ ¿µÇâÀ» ¹Þ´Â rowÀÇ °¾ö¸¦ ±¸Ñ´Ù.

Description

`int mysql_affected_rows(int [link_identifier] );`

mysql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use **mysql_num_rows()**.

mysql_close

mysql_close -- MySQL connectionÀ» ´Ý´Â´Ù.

Description

`int mysql_close(int [link_identifier] );`

Returns: true on success, false on error

mysql_close() closes the link to a MySQL database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

mysql_close() will not close persistent links generated by mysql_pconnect().

See also: **mysql_connect()**, and **mysql_pconnect()**.

mysql_connect

mysql_connect -- MySQL server connectionÀ» ¿¬´Ù.

Description

`int mysql_connect(string [hostname] , string [username] , string [password] );`

Returns: A positive MySQL link identifier on success, or false on error.

mysql_connect() establishes a connection to a MySQL server. All of the arguments are optional, and if they're missing, defaults are assumed ('localhost', user name of the user that owns the server process, empty password). The hostname string can also include a port number. eg. "hostname:port"

In case a second call is made to **mysql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **mysql_close()**.

See also **mysql_pconnect()**, and **mysql_close()**.

mysql_create_db

mysql_create_db - - MySQL database¸¦ ¸¸µç´Ù.

Description

int mysql_create_db(string database name, int [link_identifier]);

mysql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: **mysql_drop_db()**. For downwards compatibility **mysql_createdb()** can also be used.

mysql_data_seek

mysql_data_seek - - internal row pointer¸¦ ¿Å±ä´Ù.

Description

int mysql_data_seek(int result_identifier, int row_number);

Returns: true on success, false on failure

mysql_data_seek() moves the internal row pointer of the MySQL result associated with the specified result identifier to pointer to the specifyed row number. The next call to **mysql_fetch_row()** would return that row.

See also: **mysql_data_seek()**.

mysql_dbname

mysql_dbname - - ÇöÀç MySQL databaseÀÇ À̸§À» ±¸Ñ´Ù.

Description

string mysql_dbname(string result, int i);

mysql_dbname() returns the database name stored in position *i* of the result pointer returned from the **mysql_list_dbs()** function. The **mysql_num_rows()** function can be used to determine how many database names are available.

mysql_db_query

mysql_db_query - - MySQL ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

int mysql_db_query(string database, string query, int link_identifier);

Returns: A positive MySQL result identifier to the query result, or false on error.

mysql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the MySQL server and if no such link is found it'll try to create one as if **mysql_connect()** was called with no arguments

See also **mysql_connect()**. For downwards compatibility **mysql()** can also be used.

mysql_drop_db

mysql_drop_db - - MySQL database¸¦ ¹ö¸°´Ù.(drop = delete)

Description

int mysql_drop_db(string database_name, int [link_identifier]);

Returns: true on success, false on failure.

mysql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: **mysql_create_db()**. For downward compatibility **mysql_dropdb()** can also be used.

mysql_errno

mysql_errno - - ¸¶Áö¸· MySQL È£ÃâÀÇ ¿¡·¯ ¹øÈ£¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int mysql_errno(int [link_identifier]);

Errors coming back from the mySQL database backend no longer issue warnings. Instead, use these functions to retrieve the error number.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

See also: **mysql_error()**

mysql_error

mysql_error - - ¸¶Áö¸· MySQL È£ÃâÀÇ ¿¡·¯ ¸Þ¼¼Áö¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

string mysql_error(int [link_identifier]);

Errors coming back from the mySQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

See also: **mysql_errno()**

mysql_fetch_array

mysql_fetch_array - - row¸¦ ¹è¿·Î °¡Á®¿Â´Ù.

Description

array mysql_fetch_array(int result);

Returns an array that corresponds to the fetched row, or false if there are no more rows.

mysql_fetch_array() is an extended version of **mysql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **mysql_fetch_array()** is NOT significantly slower than using **mysql_fetch_row()**, while it provides a significant added value.

For further details, also see **mysql_fetch_row()**
Example 1. mysql fetch array

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
```

```
while($row = mysql_fetch_array($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result($result);
?>
```

mysql_fetch_field

mysql_fetch_field - - ÇÊµå Á¤º¸¸¦ ±¸ÇÑ´Ù.

Description

object mysql_fetch_field(int result, int [field_offset]);

Returns an object containing field information.

mysql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **mysql_fetch_field()** is retreived.

The properties of the object are:

name - column name
table - name of the table the column belongs to
max_length - maximum length of the column
not_null - 1 if the column cannot be null
primary_key - 1 if the column is a primary key
unique_key - 1 if the column is a unique key
multiple_key - 1 if the column is a non-unique key
numeric - 1 if the column is numeric
blob - 1 it the column is a BLOB
type - the type of the column
unsigned - 1 if the column is unsigned
zerofill - 1 if the column is zero-filled

See also **mysql_field_seek()**

mysql_fetch_lengths

mysql_fetch_lengths - - output columnÀÇ ÃÖ´ë data ±æÀÌ¸¦ ±¸ÇÑ´Ù.

Description

int mysql_fetch_lengths(int result);

Returns: An array that corresponds to the lengths of each field in the last row fetched by **mysql_fetch_row()**, or false on error.

mysql_fetch_lengths() stores the lengths of each result column in the last row returned by **mysql_fetch_row()** in an array, starting at offset 0.

See also: **mysql_fetch_row()**.

mysql_fetch_object

mysql_fetch_object - - row¸¦ °´Ã¼(Object)·Î °¡Á®¿Â´Ù.

Description

int mysql_fetch_object(int result);

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

mysql_fetch_object() is similar to **mysql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **mysql_fetch_array()**, and almost as quick as **mysql_fetch_row()** (the difference is insignificant).
Example 1. mysql fetch object

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

See also: **mysql_fetch_array()** and **mysql_fetch_row()**.

mysql_fetch_row

mysql_fetch_row - - row¸¦ ¹è¿-(enumerated array)·Î °¡Á®¿Â´Ù.

Description

array mysql_fetch_row(int result);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mysql_fetch_row()** would return the next row in the result set, or false if there are no more rows.

See also: **mysql_fetch_array()**, **mysql_fetch_object()**, **mysql_data_seek()**, **mysql_fetch_lengths()**, and **mysql_result()**.

mysql_field_name

mysql_field_name - - ÇÊµå ÀÌ¸§À» ±¸Ñ´Ù.

Description

string mysql_field_name(string result, int i);

mysql_field_name() returns the name of the specified field. Arguments to the function is the result identifier and the field index, ie. `mysql_field_name($result,2);`

Will return the name of the second field in the result associated with the result identifier.

For downwards compatibility **mysql_fieldname()** can also be used.

mysql_field_seek

mysql_field_seek - - ÇÊµåÀÇ offsetÀ» ¼³Á¤ÇÑ´Ù.

Description

int mysql_field_seek(int result, int field_offset);

Seeks to the specified field offset. If the next call to **mysql_fetch_field()** won't include a field offset, this field would be returned.

See also: **mysql_fetch_field()**.

mysql_field_table

mysql_field_table - - ÇØ´ç ÇÊµå¸¦ °¡Á® Table ÀÌ¸§À» ±¸Ñ´Ù.

Description

string mysql_field_table(int result, int field_offset);

Get the table name for field. For downward compatibility **mysql_fieldtable()** can also be used.

mysql_field_type

mysql_field_type - - ÇÊµå typeÀ» ±¸ÇÑ´Ù.

Description

string mysql_field_type(string result, int field_offset);

mysql_field_type() is similar to the **mysql_field_name()** function. The arguments are identical, but the field type is returned. This will be one of "int", "real", "string", "blob", or others as detailed in the MySQL documentation.
Example 1. mysql field types

```
<?php
mysql_connect("localhost:3306");
mysql_select_db("wisconsin");
$result = mysql_query("SELECT * FROM onek");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$i = 0;
$table = mysql_field_table($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = mysql_field_type  ($result, $i);
    $name  = mysql_field_name  ($result, $i);
    $len   = mysql_field_len   ($result, $i);
    $flags = mysql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
mysql_close();
?>
```

For downward compatibility **mysql_fieldtype()** can also be used.

mysql_field_flags

mysql_field_flags - - ÇÊµå flag¸¦ ±¸ÇÑ´Ù.

Description

string mysql_field_flags(string result, int field_offset);

mysql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using **explode()**.

The following flags are reported, if your version of MySQL is current enough to support them: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

For downward compatibility **mysql_fieldflags()** can also be used.

mysql_field_len

mysql_field_len - - ÇÊµå ±æÀ̸¦ ±¸ÇÑ´Ù.

Description

int mysql_field_len(string result, int field_offset);

mysql_field_len() returns the length of the specified field. For downward compatibility **mysql_fieldlen()** can also be used.

mysql_free_result

mysql_free_result - - result memory¸¦ Ç®¾îÁØ´Ù.

Description

int mysql_free_result(int result);

mysql_free_result() only needs to be called if you are worried about using too much memory while your script is running. All associated result memory for the specified result identifier will automatically be freed.

For downward compatibility **mysql_freeresult()** can also be used.

mysql_insert_id

mysql_insert_id - - ÃÖ±Ù INSERT¿¡¼- ¸¸µé¾îÁø id (generated id)¸¦ ±¸ÇÑ´Ù.

Description

int mysql_insert_id(int [link_identifier]);

mysql_insert_id() returns the ID generated for an AUTO_INCREMENTED field. This function takes no arguments. It will return the auto - generated ID returned by the last INSERT query performed.

mysql_list_fields

mysql_list_fields - - result fieldµéÀ» ³ª¿-ÇÑ´Ù.

Description

int mysql_list_fields(string database, string tablename);

mysql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **mysql_field_flags()**, **mysql_field_len()**, **mysql_field_name()**, and **mysql_field_type()**.

A result identifier is a positive integer. The function returns - 1 if a error occurs. A string describing the error will be placed in $phperrmsg, and unless the function was called as @mysql() then this error string will also be printed out.

For downward compatibility **mysql_listfields()** can also be used.

mysql_list_dbs

mysql_list_dbs - - serverÀÇ MySQL databaseµéÀ» ³ª¿-ÇÑ´Ù.

Description

int mysql_listdbs(void);

mysql_listdbs() will return a result pointer containing the databases available from the current mysql daemon. Use the **mysql_dbname()** function to traverse this result pointer.

For downward compatibility **mysql_listdbs()** can also be used.

mysql_list_tables

mysql_list_tables - - MySQL databaseÀÇ tableµéÀ» ³ª¿-ÇÑ´Ù.

Description

int mysql_list_tables(string database);

mysql_list_tables() takes a database name and result pointer much like the **mysql_db_query()** function. The **mysql_tablename()** function should be used to extract the actual table names from the result pointer.

For downward compatibility **mysql_listtables()** can also be used.

mysql_num_fields

mysql_num_fields - - resultÀÇ field °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int mysql_num_fields(int result);

mysql_num_fields() returns the number of fields in a result set.

See also: **mysql_db_query()**, **mysql_query()**, **mysql_fetch_field()**, **mysql_num_rows()**.

For downward compatibility **mysql_numfields()** can also be used.

mysql_num_rows

mysql_num_rows - - resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int mysql_num_rows(string result);

mysql_num_rows() returns the number of rows in a result set.

See also: **mysql_db_query()**, **mysql_query()** and, **mysql_fetch_row()**.

For downward compatibility **mysql_numrows()** can also be used.

mysql_pconnect

mysql_pconnect - - ¿µ±¸ÀûÀÎ MySQL Á¢¼ÓÀ» ¿¬´Ù.

Description

`int mysql_pconnect`(string [hostname] , string [username] , string [password]);

Returns: A positive MySQL persistent link identifier on success, or false on error

mysql_pconnect() acts very much like **mysql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**mysql_close()** will not close links established by `mysql_pconnect()`).

This type of links is therefore called 'persistent'.

mysql_query

mysql_query - - MySQL ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

`int mysql_query`(string query, int [link_identifier]);

mysql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **mysql_connect()** was called, and use it.

update, insert, delete ÁúÀÇÀÇ °æ¿ì´Â ´ÜÁö TRUE/FALSE¸¸ À» ¹ÝÈ¯ÇÏ°í Á¾·áÇÑ´Ù. ±×·¯³ª, selectÁúÀÇÀÇ °æ¿ì ¼°°øÇÏ¸é »õ result identifier¸¦ ¹ÝÈ¯ÇÏ±â ¶§¹®¿¡ ³ªÁß¿¡ ²À **mysql_free_result()** ¸¦ »ç¿ëÇÏ¿© »ç¿ëµÈ ÀÚ¿øÀ» ¹ÝÈ¯ÇÏ´Â °ÍÀº ÀØÁö ¸»µµ ·ÏÇÏÀÚ.

See also: **mysql_db_query()**, **mysql_select_db()**, and **mysql_connect()**.

mysql_result

mysql_result - - result data¸¦ ±¸ÇÑ´Ù.

Description

`int mysql_result`(int result, int row, mixed field);

mysql_result() returns the contents of one cell from a MySQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH

quicker than **mysql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls **mysql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: **mysql_fetch_row()**, **mysql_fetch_array()**, and **mysql_fetch_object()**.

mysql_select_db

mysql_select_db -- MySQL database¸¦ ¼±ÅÃÇÑ´Ù.

Description

int mysql_select_db(string database_name, int [link_identifier]);

Returns: true on success, false on error

mysql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mysql_connect()** was called, and use it.

Every subsequent call to **mysql_query()** will be made on the active database.

See also: **mysql_connect()**, **mysql_pconnect()**, and **mysql_query()**

For downward compatibility **mysql_selectdb()** can also be used.

mysql_tablename

mysql_tablename -- ÇØ´ç ÇÊµå¸¦ °¡Á® ¿Â Table ÀÌ¸§À» ±¸Ñ´Ù.

Description

string mysql_tablename(int result, int i);

mysql_tablename() takes a result pointer returned by the **mysql_list_tables()** function as well as an integer index and returns the name of a table. The **mysql_num_rows()** function may be used to determine the number of tables in the result pointer.

Example 1. mysql_tablename() example

```
<?php
mysql_connect ("localhost:3306");
$result = mysql_listtables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows ($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XXXI. Sybase Functions

Table of Contents

sybase_affected_rows

sybase_affected_rows - - ÃÖ±Ù ÁúÀÇ¿¡ ¿µÇâÀ» ¹Þ´Â rowÀÇ °³¼ö¸¦ ±¸Ñ´Ù.

Description

int sybase_affected_rows(int [link_identifier]);

Returns: The number of affected rows by the last query.

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use **sybase_num_rows()**.

sybase_close

sybase_close - - Sybase connectionÀ» ´Ý´Â´Ù.

Description

int sybase_close(int link_identifier);

Returns: true on success, false on error

sybase_close() closes the link to a Sybase database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non - persistent open links are automatically closed at the end of the script's execution.

sybase_close() will not close persistent links generated by sybase_pconnect().

See also: **sybase_connect()**, **sybase_pconnect()**.

sybase_connect

sybase_connect - - Sybase server connectionÀ» ¿¬´Ù.

Description

int sybase_connect(string servername, string username, string password);

Returns: A positive Sybase link identifier on success, or false on error.

sybase_connect() establishes a connection to a Sybase server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to sybase_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **sybase_close()**.

See also **sybase_pconnect()**, **sybase_close()**.

sybase_data_seek

sybase_data_seek - - internal row pointer¸¦ ¿Å±ä´Ù.

Description

int sybase_data_seek(int result_identifier, int row_number);

Returns: true on success, false on failure

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result

identifier to pointer to the specifyed row number. The next call to **sybase_fetch_row()** would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array

sybase_fetch_array - - row¸¦ ¹è¿-·Î °¡Á®¿Â´Ù.

Description

int sybase_fetch_array(int result);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

sybase_fetch_array() is an extended version of **sybase_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using sybase_fetch_array() is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

For further details, also see **sybase_fetch_row()**

sybase_fetch_field

sybase_fetch_field - - ÇÊµå Á¤º¸¸¦ ±¸Ñ´Ù.

Description

object sybase_fetch_field(int result, int field_offset);

Returns an object containing field information.

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by sybase_fetch_field() is retreived.

The properties of the object are:

name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.

column_source - the table from which the column was taken

max_length - maximum length of the column

numeric - 1 if the column is numeric

See also **sybase_field_seek()**

sybase_fetch_object

sybase_fetch_object - - row¸¦ °´Ã¼(Object)·Î °¡Á®¿Â´Ù.

Description

int sybase_fetch_object(int result);

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

sybase_fetch_object() is similar to **sybase_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **sybase_fetch_array()**, and almost as quick as **sybase_fetch_row()** (the difference is insignificant).

See also: **sybase_fetch-array()** and **sybase_fetch-row()**.

sybase_fetch_row

sybase_fetch_row - - row¸¦ ¹è¿-(enumerated array)·Î °¡Á®¿Â´Ù.

Description

array sybase_fetch_row(int result);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to sybase_fetch_rows() would return the next row in the result set, or false if there are no more rows.

See also: **sybase_fetch_array()**, **sybase_fetch_object()**, **sybase_data_seek()**, **sybase_fetch_lengths()**, and **sybase_result()**.

sybase_field_seek

sybase_field_seek - - ÇÊµåÀÇ offsetÀ» ¼³Á¤ÇÑ´Ù.

Description

int sybase_field_seek(int result, int field_offset);

Seeks to the specified field offset. If the next call to **sybase_fetch_field()** won't include a field offset, this field would be returned.

See also: **sybase_fetch_field()**.

sybase_free_result

sybase_free_result - - result memory¸¦ Ç®¾îÁØ´Ù.

Description

int sybase_free_result(int result);

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script, you may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

sybase_num_fields

sybase_num_fields - - resultÀÇ field °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int sybase_num_fields(int result);

sybase_num_fields() returns the number of fields in a result set.

See also: **sybase_db_query()**, **sybase_query()**, **sybase_fetch_field()**, **sybase_num_rows()**.

sybase_num_rows

sybase_num_rows - - resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

int sybase_num_rows(string result);

sybase_num_rows() returns the number of rows in a result set.

See also: **sybase_db_query()**, **sybase_query()** and, **sybase_fetch_row()**.

sybase_pconnect

sybase_pconnect - - ¿µ±¸ÀûÀÎ Sybase Á¢¼ÓÀ» ¿¬´Ù.

Description

int sybase_pconnect(string servername, string username, string password);

Returns: A positive Sybase persistent link identifier on success, or false on error

sybase_pconnect() acts very much like **sybase_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**sybase_close()** will not close links established by sybase_pconnect()).

This type of links is therefore called 'persistent'.

sybase_query

sybase_query - - Sybase ÁúÀÇ¸¦ Àü¼ÛÇÑ´Ù.

Description

int sybase_query(string query, int link_identifier);

Returns: A positive Sybase result identifier on success, or false on error.

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **sybase_connect()** was called, and use it.

See also: **sybase_db_query()**, **sybase_select_db()**, and **sybase_connect()**.

sybase_result

sybase_result - - result data¸¦ ±¸Ñ´Ù.

Description

int sybase_result(int result, int i, mixed field);

Returns: The contents of the cell at the row and offset in the specified Sybase result set.

sybase_result() returns the contents of one cell from a Sybase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high - performance alternatives: **sybase_fetch_row()**, **sybase_fetch_array()**, and **sybase_fetch_object()**.

sybase_select_db

sybase_select_db - - Sybase database¸¦ ¼±ÅÃÇÑ´Ù.

Description

int sybase_select_db(string database_name, int link_identifier);

Returns: true on success, false on error

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **sybase_connect()** was called, and use it.

Every subsequent call to **sybase_query()** will be made on the active database.

See also: **sybase_connect()**, **sybase_pconnect()**, and **sybase_query()**

XXXII. Network Functions

Table of Contents

fsockopen

fsockopen - - InternetÀÌ³ª Unix domainÀÇ socket connectionÀ» ¿¬´Ù.

Description

`int fsockopen`(string hostname, int port, int [errno], string [errstr]);

Opens an Internet domain socket connection to *hostname* on port *port* and returns a file pointer, which may be used by **fgets()**, **fgetss()**, **fputs()**, and **fclose()**. If the call fails, it will return FALSE and if the optional *errno* and *errstr* arguments are present they will be set to indicate the actual system level error that ocurred on the system - level connect() call. If the returned errno is 0, but the function returned FALSE, it is an indication that the error occurred before the connect() call. This is most likely due to a problem initializing the socket. Note that the errno and errstr arguments should be passed by reference.

If *port* is 0 and the operating system supports Unix domain sockets, *hostname* will be used as the filename of a Unix domain socket to connect to.

The socket will by default be opened in blocking mode. You can switch it to non - blocking mode by using the **set_socket_blocking()**.
Example 1. fsockopen example

```
$fp = fsockopen("www.php.net", 80, &$errno, &$errstr);
if(!$fp) {
        echo "$errstr ($errno)<br>\n";
} else {
        while(!feof($fp)) {
                echo fgets($fp,128);
        }
        fclose($fp);
}
```

pfsockopen

pfsockopen - - ¿µ±¸ÀûÀÎ(persistent) InternetÀÌ³ª Unix domainÀÇ socket ¿¬°áÀ» ¿¬´Ù.

Description

`int pfsockopen`(string hostname, int port, int [errno], string [errstr]);

This function behaves exactly as **fsockopen()** with the difference that the connection is not closed after the request is finished. It is the persistent version of fsockopen.

set_socket_blocking

set_socket_blocking - - ¼ÒÄÏÀÇ blocking/non - blocking ¸ðµå¸¦ ¼³Á¤ÇÑ´Ù.

Description

int set_socket_blocking(int socket descriptor, int mode);

If *mode* is false, the given socket descriptor will be switched to non-blocking mode, and if true, it will be switched to blocking mode. This affects calls like **fgets()** that read from the socket. In non-blocking mode an fgets() call will always return right away while in blocking mode it will wait for data to become available on the socket.

gethostbyaddr

gethostbyaddr - - ÁÖ¾îÁø IP address¿¡ ÇØ´çÇÏ´Â Internet host nameÀ» ±¸ÇÑ´Ù.

Description

string gethostbyaddr(string ip_address);

Returns the host name of the Internet host specified by *ip_address*. If an error occurs, returns *ip_address*.

See also **gethostbyname()**.

gethostbyname

gethostbyname - - ÁÖ¾îÁø Internet host name¿¡ ÇØ´çÇÏ´Â IP address¸¦ ±¸ÇÑ´Ù.

Description

string gethostbyname(string hostname);

Returns the IP address of the Internet host specified by *hostname*.

See also **gethostbyaddr()**.

gethostbynamel

gethostbynamel - - ÁÖ¾îÁø Internet host name¿¡ ÇØ´çÇÏ´Â IP addresseµéÀÇ list¸¦ ±¸ÇÑ´Ù.

Description

array gethostbynamel(string hostname);

Returns a list of IP addresses to which the Internet host specified by *hostname* resolves.

See also **gethostbyname()**, **gethostbyaddr()**, **checkdnsrr()**, **getmxrr()**, and the named(8) manual page.

checkdnsrr

checkdnsrr - - ÁÖ¾îÁø Internet host name or IP address¿¡ ÇØ´çÇÏ´Â DNS recordµéÀ» °Ë»çÇÑ´Ù .

Description

int checkdnsrr(string host, string [type]);

Searches DNS for records of type *type* corresponding to *host*. Returns true if any records are found; returns false if no records were found or if an error occurred.

type may be any one of: A, MX, NS, SOA, PTR, CNAME, or ANY. The default is MX.

host may either be the IP address in dotted-quad notation or the host name.

See also **getmxrr()**, **gethostbyaddr()**, **gethostbyname()**, **gethostbynamel()**, and the named(8) manual page.

getmxrr

getmxrr - - ÁÖ¾îÁø Internet host name¿¡ ÇØ´çÇÏ´Â MX recordµéÀ» ±¸ÇÑ´Ù.

Description

int getmxrr(string hostname, array mxhosts, array [weight]);

Searches DNS for MX records corresponding to *hostname*. Returns true if any records are found; returns false if no records were found or if an error occurred.

A list of the MX records found is placed into the array *mxhosts*. If the *weight* array is given, it will be filled with the weight information gathered.

See also **checkdnsrr()**, **gethostbyname()**, **gethostbynamel()**, **gethostbyaddr()**, and the named(8) manual page.

openlog

openlog - - system logger·ÎÀÇ connectionÀ» ¿¬´Ù.

Description

void openlog(string ident, int option, int facility);

openlog() opens a connection to the system logger for a program. The string *ident* is added to each message. Values for *option* and *facility* are given in the next section. The use of openlog() is optional; It will automatically be called by **syslog()** if necessary, in which case ident will default to false. See also **syslog()** and **closelog()**.

syslog

syslog - - system log message¸¦ ¸¸µç´Ù.

Description

void syslog(int priority, string message);

syslog() generates a log message that will be distributed by the system logger. *priority* is a combination of the facility and the level, values for which are given in the next section. The remaining argument is the message to send, except that the two characters %m will be replaced by the error message string (strerror) corresponding to the present value of errno.

More information on the syslog facilities can be found in the man pages for syslog on Unix machines.

On Windows NT, the syslog service is emulated using the Event Log.

closelog

closelog - - system logger·ÎÀÇ connectionÀ» ´Ý´Â´Ù.

Description

string passthru(void);

closelog() closes the descriptor being used to write to the system logger. The use of **closelog()** is optional.

debugger_on

debugger_on - - ³»ÀåµÈ PHP debugger¸¦ enable ÇÑ´Ù.

Description

void debugger_on(string address);

Enables the internal PHP debugger, connecting it to *address*. The debugger is still under development.

debugger_off

debugger_off - - ³»ÀåµÈ PHP debugger¸¦ disable ÇÑ´Ù.

Description

void debugger_off(void);

Disables the internal PHP debugger. The debugger is still under development.

XXXIII. ODBC Functions

Table of Contents

odbc_autocommit

odbc_autocommit - - autocommit ±â´ÉÀ» ÄÑ°í, ²ö´Ù.

Description

`int odbc_autocommit`(int connection_id, int [OnOff]);

Without the *`OnOff`* parameter, this function returns auto - commit status for *`connection_id`*. True is returned if auto - commit is on, false if it is off or an error occurs.

If *`OnOff`* is true, auto - commit is enabled, if it is false auto - commit is disabled. Returns `true` on success, `false` on failure.

By default, auto - commit is on for a connection. Disabling auto - commit is equivalent with starting a transaction.

See also **odbc_commit()** and **odbc_rollback()**.

odbc_binmode

odbc_binmode - - binary column data¸¦ ´Ù·é´Ù.

Description

`int odbc_binmode`(int result_id, int mode);

(ODBC SQL types affected: BINARY, VARBINARY, LONGVARBINARY)

ODBC_BINMODE_PASSTHRU: Passthru BINARY data

ODBC_BINMODE_RETURN: Return as is

ODBC_BINMODE_CONVERT: Convert to char and return

When binary SQL data is converted to character C data, each byte (8 bits) of source data is represented as two ASCII characters. These characters are the ASCII character representation of the number in its hexadecimal form. For example, a binary 00000001 is converted to `"01"` and a binary 11111111 is converted to `"FF"`.

Table 1. LONGVARBINARY handling

binmode	longreadlen	result
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	> 0	passthru
ODBC_BINMODE_RETURN	> 0	return as is
ODBC_BINMODE_CONVERT	> 0	return as char

If **odbc_fetch_into()** is used, passthru means that an empty string is returned for these columns.

If *result_id* is `0`, the settings apply as default for new results.

> **Note:** Default for longreadlen is `4096` and binmode defaults to `ODBC_BINMODE_RETURN`. Handling of binary long columns is also affected by **odbc_longreadlen()**

odbc_close

odbc_close - - ODBC connectionÀ» ´Ý´Â´Ù.

Description

void odbc_close(int connection_id);

odbc_close() will close down the connection to the database server associated with the given connection identifier.

> **NOTE:** This function will fail if there are open transactions on this connection. The connection will remain open in this case.

odbc_close_all

odbc_close_all - - ¸ðµç ODBC connectionµéÀ» ´Ý´Â´Ù.

Description

void odbc_close_all(void);

odbc_close_all() will close down all connections to database server(s).

> **NOTE:** This function will fail if there are open transactions on a connection. This connection will remain open in this case.

odbc_commit

odbc_commit - - ODBC transactionÀ» commitÇÑ´Ù.

Description

int odbc_commit(int connection_id);

Returns: `true` on success, `false` on failure. All pending transactions on *connection_id* are committed.

odbc_connect

odbc_connect - - datasource¿Í ¿¬°áÇÑ´Ù.

Description

int odbc_connect(string dsn, string user, string password);

Returns an ODBC connection id or 0 (`false`) on error.

The connection id returned by this functions is needed by other ODBC functions. You can have multiple connections open at once. For persistent connections see **odbc_pconnect()**.

odbc_cursor

odbc_cursor - - cursornameÀ» ±¸ÇÑ´Ù.

Description

string odbc_cursor(int result_id);

odbc_cursor will return a cursorname for the given result_id.

odbc_do

odbc_do - - **odbc_exec()**¿Í °°´Ù.

Description

string odbc_do(int conn_id, string query);

odbc_do will execute a query on the given connection

odbc_exec

odbc_exec - - SQL ¹®ÀåÀ» ÁØºñÇÏ°í ½ÇÇàÇÑ´Ù.

Description

int odbc_exec(int connection_id, string query_string);

Returns `false` on error. Returns an ODBC result identifier if the SQL command was executed successfully.

odbc_exec() will send an SQL statement to the database server specified by *`connection_id`*. This parameter must be a valid identifier returned by **odbc_connect()** or **odbc_pconnect()**.

See also: **odbc_prepare()** and **odbc_execute()** for multiple execution of SQL statements.

odbc_execute

odbc_execute - - ½ÇÇàÇÒ ÁØºñ°¡ µÈ ¹®ÀåÀ» ½ÇÇàÇÑ´Ù.

Description

`int odbc_execute`(int result_id, array [parameters_array]);

Executes a statement prepared with **odbc_prepare()**. Returns `true` on successful execution, `false` otherwise. The array *`arameters_array`* only needs to be given if you really have parameters in your statement.

odbc_fetch_into

odbc_fetch_into - - ÇÑ°³ÀÇ result row¸¦ ¹è¿·Î °¡Á®¿Â´Ù.

Description

`int odbc_fetch_into`(int result_id, int [rownumber], array result_array);

Returns the number of columns in the result; `false` on error. *`result_array`* must be passed by reference, but it can be of any type since it will be converted to type array. The array will contain the column values starting at array index 0.

odbc_fetch_row

odbc_fetch_row - - row¸¦ °¡Á®¿Â´Ù.

Description

`int odbc_fetch_row`(int result_id, int [row_number]);

If **odbc_fetch_row()** was succesful (there was a row), `true` is returned. If there are no more rows, `false` is returned.

odbc_fetch_row() fetches a row of the data that was returned by **odbc_do()** / **odbc_exec()**. After **odbc_fetch_row()** is

called, the fields of that row can be accessed with **odbc_result()**.

If *row_number* is not specified, **odbc_fetch_row()** will try to fetch the next row in the result set. Calls to **odbc_fetch_row()** with and without *row_number* can be mixed.

To step through the result more than once, you can call **odbc_fetch_row()** with *row_number* 1, and then continue doing **odbc_fetch_row()** without *row_number* to review the result. If a driver doesn't support fetching rows by number, the *row_number* parameter is ignored.

odbc_field_name

odbc_field_name - - columnnameÀ» ±¸ÇÑ´Ù.

Description

string odbc_fieldname(int result_id, int field_number);

odbc_field_name() will return the name of the field occupying the given column number in the given ODBC result identifier. Field numbering starts at 1. `false` is returned on error.

odbc_field_type

odbc_field_type - - fieldÀÇ datatypeÀ» ±¸ÇÑ´Ù.

Description

string odbc_field_type(int result_id, mixed field);

odbc_field_type() will return the SQL type of the field referecend by name or number in the given ODBC result identifier. Field numbering runs from 1.

odbc_field_len

odbc_field_len - - fieldÀÇ ±æÀÌ¸¦ ±¸ÇÑ´Ù.

Description

`string odbc_field_type`(int result_id, int field_number);

odbc_field_type() will return the length of the field referecend by number in the given ODBC result identifier. Field numbering starts at 1.

odbc_free_result

odbc_free_result - - ÁöÁ¤µÈ result¿¡ °ü·ÃµÈ resource¸¦ ÇØÁ¦ÇÑ´Ù.

Description

int odbc_free_result(int result_id);

Always returns `true`.

odbc_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **odbc_free_result()**, and the memory associated with *result_id* will be freed.

> **NOTE:** If auto - commit is disabled (see **odbc_autocommit()**) and you call **odbc_free_result()** before commiting, all pending transactions are rolled back.

odbc_longreadlen

odbc_longreadlen - - LONG columnµéÀ» ´Ù·é´Ù.

Description

int odbc_longreadlen(int result_id, int length);

(ODBC SQL types affected: LONG, LONGVARBINARY) The number of bytes returned to PHP is controled by the parameter length. If it is set to 0, Long column data is passed thru to the client.

NOTE: Handling of LONGVARBINARY columns is also affected by **odbc_binmode()**

odbc_num_fields

odbc_num_fields - - resultÀÇ column °³¼ö

Description

int odbc_num_fields(int result_id);

odbc_num_fields() will return the number of fields (columns) in an ODBC result. This function will return -1 on error. The argument is a valid result identifier returned by **odbc_exec()**.

odbc_pconnect

odbc_pconnect - - ¿µ±¸ÀûÀÎ database connectionÀ» ¿¬°Ù.

Description

int odbc_pconnect(string dsn, string user, string password);

Returns an ODBC connection id or 0 (false) on error. This function is much like **odbc_connect()**, except that the connection is not really closed when the script has finished. Future requests for a connection with the same *dsn*, *user*, *password* combination (via **odbc_connect()** and **odbc_pconnect()**) can reuse the persistent connection.

NOTE: Persistent connections have no effect if PHP is used as a CGI program.

For more information on persistent connections, refer to the PHP3 FAQ.

odbc_prepare

odbc_prepare - - ½ÇÇàÇÒ ¹®ÀåÀ» ÁØºñÇÑ´Ù.

Description

int odbc_prepare(int connection_id, string query_string);

Returns false on error.

Returns an ODBC result identifier if the SQL command was prepared successfully. The result identifier can be used later to execute the statement with **odbc_execute()**.

odbc_num_rows

odbc_num_rows - - resultÀÇ row °³¼ö

Description

int odbc_num_rows(int result_id);

odbc_num_rows() will return the number of rows in an ODBC result. This function will return -1 on error. For INSERT, UPDATE and DELETE statements **odbc_num_rows()** returns the number of rows affected. For a SELECT clause this can be the number of rows available.

Note: Using **odbc_num_rows()** to determine the number of rows available after a SELECT will return -1 with many drivers.

odbc_result

odbc_result - - result data¸¦ ±¸ÇÑ´Ù.

Description

string odbc_result(int result_id, mixed field);

Returns the contents of the field.

field can either be an integer containing the column number of the field you want; or it can be a string containing the name of the field. For example:

```
$item_3 = odbc_result($Query_ID, 3 );
$item_val = odbc_result($Query_ID, "val");
```

The first call to **odbc_result()** returns the value of the third field in the current record of the query result. The second function call to **odbc_result()** returns the value of the field whose field name is "val" in the current record of the query result. An error occurs if a column number parameter for a field is less than one or exceeds the number of columns (or fields) in the current record. Similarly, an error occurs if a field with a name that is not one of the fieldnames of the table(s) that is(are) being queried.

Field indices start from 1. Regarding the way binary or long column data is returned refer to **odbc_binmode()** and **odbc_longreadlen()**.

odbc_result_all

odbc_result_all - - result¸¦ HTML table·Î ÀÎ¼âÇÑ´Ù.

Description

`int odbc_result_all`(int result_id, string [format]);

Returns the number of rows in the result or `false` on error.

odbc_result_all() will print all rows from a result identifier produced by **odbc_exec()**. The result is printed in HTML table format. With the optional string argument *format*, additional overall table formatting can be done.

odbc_rollback

odbc_rollback - - transactionÀ» Rollback ÇÑ´Ù.

Description

int odbc_rollback(int connection_id);

Rolls back all pending statements on *connection_id*. Returns `true` on success, `false` on failure.

odbc_setoption

odbc_setoption - - ODBC ¼³Á¤À» Á¶Á¤ÇÑ´Ù. ¿¡·¯°¡ ¹ß»ýÇÑ °æ¿ì false¸¦ ¹ÝÈ¯ÇÏ°í ¼º°øÀ¸·Î true¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

`int odbc_setoption`(int id, int function, int option, int param);

This function allows fiddling with the ODBC options for a particular connection or query result. It was written to help find work arounds to problems in quirky ODBC drivers. You should probably only use this function if you are an ODBC programmer and understand the effects the various options will have. You will certainly need a good ODBC reference to explain all the different options and values that can be used. Different driver versions support different options.

Because the effects may vary depending on the ODBC driver, use of this function in scripts to be made publicly available is strongly discouraged. Also, some ODBC options are not available to this function because they must be set before the connection is established or the query is prepared. However, if on a particular job it can make PHP work so your boss doesn't tell you to use a commercial product, that's all that really matters.

Id is a connection id or result id on which to change the settings.For SQLSetConnectOption(), this is a connection id. For SQLSetStmtOption(), this is a result id.

function is the ODBC function to use. The value should be 1 for SQLSetConnectOption() and 2 for SQLSetStmtOption().

Parmeter *option* is the option to set.

Parameter *param* is the value for the given *option*.
Example 1. ODBC Setoption Examples

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//    Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//    This example has the same effect as
//    odbc_autocommit($conn, true);
odbc_setoption ($conn, 1, 102, 1);
// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//    This example sets the query to timeout after 30 seconds.
$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

XXXIV. Oracle 8 functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº Oracle8°ú Oracle7 µ¥ÀÌÅÍº£ÀÌ½º¸¦ Á¢±ÙÇÒ ¼ö ÀÖµµ·Ï ÇØÁØ´Ù. ÀÌ°ÍÀº Oracle8 Call-Interface (OCI8)¸¦ »ç¿ëÇÑ´Ù.ÀÌ µå¶óÀÌ¹ö¸¦ »ç¿ëÇÏ·Á¸é Oracle8 client libraries°¡ ÇÊ¿äÇÏ´Ù.

ÀÌ µå¶óÀÌ¹ö´Â º¸ËëÀÇ PHP3 Ora_ µå¶óÀÌ¹öÁ¶´Ù ´õ¿í À¯¿¬ÇÏ´Ù. ÀÌ°ÍÀº PHP3ÀÇ Àü¿ª, Áö¿ª º¯¼öµéÀÇ Oracle placeholder·ÎÀÇ bindingÀ» Áö¿øÇÏ°í, full LOB¿Í FILE, ROWID¸¦ Áö¿øÇÏ¸ç, user-supplied define variableÀ» »ç¿ëÇÒ ¼ö ÀÖµµ·Ï ÇØ ÁØ´Ù.

OCIDefineByName

OCIDefineByName -- SELECT ÁßÀÇ degine-stepÀ¸·Î PHP º¯¼ö¸¦ »ç¿ëÇÑ´Ù.

Description

`int OCIDefineByName`(int stmt, string Column-Name, mixed &variable, int [type]);

OCIDefineByName() uses fetches SQL-Columns into user-defined PHP-Variables. Be careful that Oracle user ALL-UPPERCASE column-names, whereby in your select you can also write lower-case. **OCIDefineByName()** expects the *Column-Name* to be in uppercase. If you define a variable that doesn't exists in you select statement, no error will be given!

If you need to define an abstract Datatype (LOB/ROWID/BFILE) you need to allocate it first using **OCINewDescriptor()** function. See also the **OCIBindByName()** function.

Example 1. OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@digicol.de (980219) */
$conn = OCILogon("scott","tiger");
$stmt = OCIParse($conn,"select empno, ename from emp");
/* the define MUST be done BEFORE ociexecute! */
OCIDefineByName($stmt,"EMPNO",&$empno);
OCIDefineByName($stmt,"ENAME",&$ename);
OCIExecute($stmt);
while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIBindByName

OCIBindByName -- PHP º¯¼ö¸¦ Oracle Placeholder·Î bind ÇÑ´Ù.

Description

int OCIBindByName(int stmt, string ph_name, mixed &variable, intlength, int [type]);

OCIBindByName() binds the PHP variable *variable* to the Oracle placeholder *ph_name*. Whether it will be used for input or output will be determined run-time, and the necessary storage space will be allocated. The *length* paramter sets the maximum length for the bind. If you set *length* to -1 **OCIBindByName()** will use the current length of *variable* to set the maximum length.

If you need to bind an abstract Datatype (LOB/ROWID/BFILE) you need to allocate it first using **OCINewDescriptor()** function. The *length* is not used for abstract Datatypes and should be set to -1. The *type* variable tells oracle, what kind of descriptor we want to use. Possible values are: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) and OCI_B_ROWID (ROWID).

Example 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@digicol.de (980221)
  inserts 3 resords into emp, and uses the ROWID for updating the
  records just after the insert.
*/
$conn = OCILogon("scott","tiger");
$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
                                        "values (:empno,:ename) ".
                                        "returning ROWID into :rid");
$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");
$rowid = OCINewDescriptor($conn,OCI_D_ROWID);
OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);
$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);
$sal = 10000;
while (list($empno,$ename) = each($data)) {
        OCIExecute($stmt);
        OCIExecute($update);
}
$rowid->free();
OCIFreeStatement($update);
OCIFreeStatement($stmt);
$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
        var_dump($arr);
}
OCIFreeStatement($stmt);
/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCILogon

OCILogon - - Oracle°ú ¿¬°áÇÑ´Ù.

Description

int OCILogon(string username, string password, string [OCACLE_SID]);

OCILogon() returns an connection identified needed for most other OCI calls.

OCILogOff

OCILogOff - - Oracle°úÀÇ ¿¬°áÀ» ²÷´Â´Ù.

Description

int OCILogOff(int connection);

OCILogOff() closes an Oracle connection.

OCIExecute

OCIExecute - - ¸í·É¹®À» ½ÇÇàÇÑ´Ù.

Description

int OCIExecute(int statement, int [mode]);

OCIExecute() executes a previously parsed statement. (see **OCIParse()**). The optional *mode* allows you to specify the execution - mode (default is OCI_COMMIT_ON_SUCCESS). If you don't want statements to be commited automaticly specify OCI_DEFAULT as your mode.

OCICommit

OCICommit - - ¼öÇàµÇ¾ú´ø transactionµéÀ» CommitÇÑ´Ù.

Description

int OCICommit(int connection);

OCICommit() commits all outstanding statements for Oracle connection *connection*.

OCIRollback

OCIRollback - - ¼öÇàµÇ¾ú´ø transactionµéÀ» Rolls backÇÑ´Ù.

Description

int OCIRoolback(int connection);

OCICommit() rolls back all outstanding statements for Oracle connection *connection*.

OCINumRows

OCINumRows - - ¿µÇâ¹Þ´Â rowÀÇ °³¼ö¸¦ ±¸Ñ´Ù.

Description

int OCINumRows(int statement);

OCINumRows() returns the number of rows affected for eg update - statements. This funtions will not tell you the number of rows that a select will return!

OCIResult

OCIResult - - fetched row¿¡ ÀÖ´Â coulumn °ªÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

int OCIResult(int statement, mixed column);

OCIResult() returns the data for column *column* in the current row (see **OCIFetch()**).**OCIResult()** will return everything as strings except for abstract types (ROWIDs, LOBs and FILEs).

OCIFetch

OCIFetch - - result - buffer·Î ´ÙÀ½ row¸¦ °¡Á®¿Â´Ù.

Description

int OCIFetch(int statement);

OCIFetch() fetches the next row (for SELECT statements) into the internal result - buffer.

OCIFetchInto

OCIFetchInto - - result - array·Î ´ÙÀ½ row¸¦ °¡Á®¿Â´Ù.

Description

int OCIFetchInto(array &result, int [mode]);

OCIFetchInto() fetches the next row (for SELECT statements) into the *result* array. **OCIFetchInto()** will overwrite the previous content of *result*. By default *result* will contain a one - based array of all columns that are not NULL.

The *mode* parameter allows you to change the default behaviour. You can specify more than one flag by simply addig them up (eg OCI_ASSOC + OCI_RETURN_NULLS). The known flags are:
OCI_ASSOC Return an associative array.
OCI_NUM Return an numbered array starting with one. (DEFAULT)
OCI_RETURN_NULLS Return empty columns.
OCI_RETURN_LOBS Return the value of a LOB instead of the desxriptor.

OCIColumnIsNULL

OCIColumnIsNULL - - result Ä®·³ÀÌ NULLÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int OCIColumnIsNULL(int stmt, mixed column);

OCIColumnIsNULL() returns true if the returned column *col* in the result from the statement *stmt* is NULL. You can either use the column - number (1 - Based) or the column - name for the *col* parameter.

OCIColumnSize

OCIColumnSize - - result Ä®·³ Å©±â¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int OCIColumnSize(int stmt, mixed column);

OCIColumnSize() returns the size of the column as given by Oracle. You can either use the column - number (1 - Based) or the column - name for the *col* parameter.

XXXV. Oracle functions

Table of Contents

Ora_Bind

Ora_Bind - - PHP º¯¼ö¸¦ Oracle ÆÄ¶ó¹ÌÅÍ·Î bindÇÑ´Ù.

Description

int ora_bind(int cursor, string PHP variable name, string SQL parameter name, int length, int [type]);

Returns true if the bind succeeds, otherwise false. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function binds the named PHP variable with a SQL parameter. The SQL parameter must be in the form ":name". With the optional type parameter, you can define whether the SQL parameter is an in/out (0, default), in (1) or out (2) parameter. As of PHP 3.0.1, you can use the constants ORA_BIND_INOUT, ORA_BIND_IN and ORA_BIND_OUT instead of the numbers.

ora_bind must be called after **ora_parse()** and before **ora_exec()**. Input values can be given by assignment to the bound PHP variables, after calling **ora_exec()** the bound PHP variables contain the output values if available.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close

Ora_Close - - Oracle cursor¸¦ ´Ý´Â´Ù.

Description

int ora_close(int cursor);

Returns true if the close succeeds, otherwise false. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

(¿ªÀÚÁÖ : ÀÌ ÇÔ¼öº´Â ¸ðÁ¦³ª "wrong parameter count" ÀÌ³ª"true",¦ ¹ÝÈ¯ÇÑ´Ù. ±×·¯³ª C - APIÀÎ "ora_del_cursor()"´Â ¿ø·¡ ¾Æ¹«°Íµµ ¹ÝÈ¯ÇÏÁö ¾Êµµ·ÏµÇ¾î ÀÖÀ¸¹Ç·Î ¹«½ÃÇÏµµ·Ï ÇÑ´Ù.)

This function closes a data cursor opened with **ora_open()**.

Ora_ColumnName

Ora_ColumnName - - Oracle result columnÀÇ ÀÌ¸§À» ±¸ÇÑ´Ù.

Description

string Ora_ColumnName(int cursor, int column);

Returns the name of the field/column *column* on the cursor *cursor*. The returned name is in all uppercase letters.

Ora_ColumnType

Ora_ColumnType - - Oracle result columnÀÇ typeÀ» ±¸ÇÑ´Ù.

Description

string Ora_ColumnType(int cursor, int column);

Returns the Oracle data type name of the field/column *column* on the cursor *cursor*. The returned type will be one of the following:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit

Ora_Commit - - Oracle transactionÀ» commit ÇÑ´Ù.

Description

int ora_commit(int conn);

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions. This function commits an Oracle transaction. A transaction is defined as all the changes on a given connection since the last commit/rollback, autocommit was turned off or when the connection was established.

Ora_CommitOff

Ora_CommitOff - - automatic commitÀ» disable ÇÑ´Ù.

Description

int ora_commitoff(int conn);

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function turns off automatic commit after each **ora_exec()**.

Ora_CommitOn

Ora_CommitOn - - automatic commitÀ» enable ÇÑ´Ù.

Description

int ora_commiton(int conn);

This function turns on automatic commit after each **ora_exec()** on the given connection.

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Error

Ora_Error - - Oracle error message¸¦ ±¸ÇÑ´Ù.

Description

`string Ora_Error`(int cursor);

Returns an error message of the form *XXX*-*NNNNN* where *XXX* is where the error comes from and *NNNNN* identifies the error message.

The *cursor* parameter can as of PHP 3. 0.6 also be a connection id.

On UNIX versions of Oracle, you can find details about an error message like this: `$` **oerr ora 00001** 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key

Ora_ErrorCode

Ora_ErrorCode - - Oracle error code¸¦ ±¸ÇÑ´Ù.

Description

int Ora_ErrorCode(int cursor);

Returns the numeric error code of the last executed statement on the specified cursor.

The *cursor* parameter can as of PHP 3. 0.6 also be a connection id.

Ora_Exec

Ora_Exec - - Oracle cursor¿¡ ÀÖ´Â parseµÈ ¹®ÀåÀ» ½ÇÇàÇÑ´Ù.

Description

int ora_exec(int cursor);

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Fetch

Ora_Fetch - - cursor¿¡¼- dataÀÇ row¸¦ °¡Á®¿Â´Ù.

Description

int ora_fetch(int cursor);

Returns true (a row was fetched) or false (no more rows, or an error occured). If an error occured, details can be retrieved using the **ora_error()** and **ora_errorcode()** functions. If there was no error, **ora_errorcode()** will return 0. Retrieves a row of data from the specified cursor.

Ora_GetColumn

Ora_GetColumn - - °¡Á®¿Â row¿¡¼- data¸¦ ±¸ÇÑ´Ù.

Description

mixed ora_getcolumn(int cursor, mixed column);

Returns the column data. If an error occurs, False is returned and **ora_errorcode()** will return a non-zero value. Note, however, that a test for False on the results from this function may be true in cases where there is not error as well (NULL result, empty string, the number 0, the string "0"). Fetches the data for a column or function result.

Ora_Logoff

Ora_Logoff - - Oracle connectionÀ» ´Ý´Â´Ù.

Description

int ora_logoff(int connection);

Returns true on success, False on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions. Logs out the user and disconnects from the server.

Ora_Logon

Ora_Logon - - Oracle connectionÀ» ¿¬´Ù.

Description

`int ora_logon`(string user, string password);

Establishes a connection between PHP and an Oracle database with the given username and password.

Connections can be made using SQL*Net by supplying the TNS name to *user* like this:

```
$conn = Ora_Logon("user@TNSNAME", "pass");
```

If you have character data with non-ASCII characters, you should make sure that `NLS_LANG` is set in your environment. For server modules, you should set it in the server's environment before starting the server.

Returns a connection index on success, or false on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Open

Ora_Open - - Oracle cursor¸¦ ¿¬´Ù.

Description

int ora_open(int connection);

Opens an Oracle cursor associated with connection.

Returns a cursor index or False on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Parse

Ora_Parse - - SQL ¹®ÀåÀ» ÇØ¼®(parse)ÇÑ´Ù.

Description

int ora_parse(int cursor_ind, string sql_statement, int defer);

This function parses an SQL statement or a PL/SQL block and associates it with the given cursor. Returns 0 on success or -1 on error.

Ora_Rollback

Ora_Rollback - - transactionÀ» Rollback ÇÑ´Ù.

Description

int ora_rollback(int connection);

This function undoes an Oracle transaction. (See **ora_commit()** for the definition of a transaction.)

Returns true on success, false on error. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

(¿ªÀÚÁÖ : ÀÌ ÀÌÇÏÀÇ Oracle °Î°ÐÀº ¿µ¹® ¸Å´º¾ó¿¡ ¾ø´Â °Î°ÐÀ¸·Î PHP È¯ÆäÀÌÁöÀÇ Note¸¦ Âü°íÇÏ¿´´Ù.)

´ÙÀ½°ú °°À° ÇÔ¼öµéÀÌ PHP3.0.2a°ÎÅÍ Ãß°¡µÇ¾ú´Ù.

Ora_Do($conn, $query)
Ora_Do()´Â ¸¹ÀÌ »ç¿ëÇÏ´Â 1È¸¼º ÁúÀÇ¸¦ À§ÇØ Ora_Parse(), Ora_Exec(), Ora_Fetch()ÀÇ µ¿ÀÛÀ» ÇÑ²¨¹ø¿¡ ¼öÇàÇÏ´Â ÇÔ¼öÀÌ´Ù.

Ora_FetchInto($curs, &$ary)
Ora_FetchInto´Â ÀüÃ¼ row¸¦ fetchÇÏ¿© °á°ú¸¦ Àü´Þ¹ÞÀº array reference¿¡ ÀúÅåÇÑ´Ù. ÀÌ ÇÔ¼ö´Â ÇØ´ç rowÀÇ ÄÃ·³ °³¼ö³ª false¸¦ ¹ÝÈ¯ÇÑ´Ù.

ora_columnsize($curs, $colindex)
ÀÌ ÇÔ¼ö´Â named column¿¡¼ type(ora_columntype()À» °¸¶ó)ÀÇ Å©±â¸¦ ¹ÝÈ¯ÇÑ´Ù. ÄÃ·³ ¹øÈ£´Â 0°ÎÅÍ ½ÃÀÛÇÑ´Ù.

ora_numcols($curs)
resultÀÇ ÄÃ·³ °³¼ö¸¦ ¹ÝÈ¯ÇÑ´Ù. ÀÌ ÇÔ¼ö´Â ora_parse()/ora_exec()/ora_fetch()ÀÇ ÇÔ¼ö°¡ ¼ø¼´ë·Î ¿Ï·áµÇ¾úÀ» ¶§¸¸ Á¤»óÀûÀÎ °ªÀ» µ¹·ÁØ´Ù. (ora_do()¸¦ ½ÇÇàÇÑ ÈÄµµ µÈ´Ù.)

ora_numrows($curs)
ÄÃ·³ °³¼ö°¡ ¾Æ´Ï¶ó rowÀÇ °³¼ö¶ó´Â °Í¸¸ Á¦¿ÜÇÏ°í´Â À§¿¡ ÀÖ´Â **ora_numcols()**¿Í °°´Ù.

ora_plogon()
¿µ±¸ÀûÀÎ ¿¬°á(persistent connection)À» ¸¸µç´Ù´Â Á¦À» Á¦¿ÜÇÏ°í´Â **ora_login()**°ú µ¿ÀÏÇÏ´Ù. ¿µ±¸Àû ¿¬°áÀº mod_php¿¡¼¸¸ ÀÇ¹Ì°¡ ÀÖ°í CGI PHP¿¡¼´Â ¾Æ¹«·± ÀÇ¹Ì°¡ ¾ø´Ù. ÀÌ°ÍÀº ¸Þ¸ð¸®¸¦ Á¶±Ý ´õ »ç¿ëÇÏ´Â ´ë½Î ¿©·¯ °ÐÀÇ mod_phpÀÇ µ¥ÀÌÅÍº£ÀÌ½º ¿¬°áÀ» ¾à°£ °ü¸®Ô ÇØ ÁØ´Ù. ÀÚ¼¼ÇÑ ³»¿ëÀº **mysql_pconnect()**¸¦ Âü°í¶ÇÏ¶ó

Oracle Example

Prerequisites

´ÙÀ½ÀÇ È¯°æ¯¼öÀÇ °ªÀ» ¾Ë°í ÀÖ¾î¾ß ÇÑ´Ù. :

ORACLE_HOME
OracleÀÌ ¼³Ä¡µÇ¾î ÀÖ´Â µð·ºÅä¸®ÀÇ path. ÀÌ ¯¼ö´Â °¸Ë ÇØ´ç ½Ã½ºÅÛÀÇ oracle »ç¿ëÀÚ·Î ¼³Á¤µÈ »ç¿ëÀÚµéÀº login ½°Å©¸³Æ®¿¡¼ ¼³Á¤µÈ´Ù.

ORACLE_SID
¿¬°áÇÏ±â¸¦ ¿øÇÏ´Â databaseÀÇ ÀÌ§. ÀÌ ¯¼ö´Â °¸Ë »ç¿ëÀÚµéÀº UNIX È¯°æ¿¡¼ ¼³Á¤µÈ´Ù.

ÀÌ ¯¼ö °ªµéÀº database¿¡ ¿¬°áÇÏ¿© sqlplus¸¦ »ç¿ëÇÒ ¼ö ÀÖ´Â »ç¿ëÀÚ·Î loginÇÏ¿© È®ÀÎÇÒ ¼ö ÀÖ´Ù. UNIX shell¿¡¼ ´ÙÀ½°ú °°ÀÌ È®ÀÎÇÑ´Ù. :

```
prompt> echo $ORACLE_HOME
/opt/oracle/oracle/8.0.3
prompt> echo $ORACLE_SID
```

ORACLE

ora_* ÇÔ¼öµéÀ» »ç¿ëÇÏ´Â °£´ÜÇÑ PHP ½ºÅ©¸³Æ® (A simple PHP script using ora_* functions)

```
<?php
  putenv("ORACLE_SID=ORACLE");
  putenv("ORACLE_HOME=/opt/oracle/oracle/8.0.3");
  $conn = ora_login("scott", "tiger");
  $curs = ora_open($conn);
  ora_commitoff($conn);
  $query = sprintf(&quotselect * from cat");
/* Long version */
/*
  ora_parse($curs, $query);
  ora_exec($curs);
  ora_fetch($curs);
*/
/* Short Version */
  ora_do($conn, $query);

  $ncols = ora_numcols($curs);
  $nrows = ora_numrows($curs);
  printf("Result size is $ncols cols by $nrows rows.<br>");
  for ($i=0; $i<$ncols; $i++) {
    printf("col[%s] = %s type[%d] = %s<br>",$i, ora_columnname($curs, $I), $i, ora_columntype($curs, $I));
  }
  for ($j=0; $j<$nrows; $j++) {
     for ($i=0; $i<$ncols; $i++) {
        $col = ora_getcolumn($curs, $I);
        printf("val[%d, %d] = %s * ", $j, $i, ora_getcolumn($curs, $I);
     }
     printf("<br>");
  }
?>
```

ORACLE_HOMEÀÌ /opt/oracle/oracle/8.0.3ÀÌ¶ó°í °¡Á¤ÇßÀ» ¶§, ¼Ö¶ó¸®½º°¿¡¼ PHP3¸¦ OracleÀ» Áö¿øÇÏµµ·Ï ¼³Á¤ÇÏ´Â ¹æ¹ýÀº ´ÙÀ½°ú °°´Ù. :

```
./configure --with-oracle \
            --with-config-file-path=/opt/GNUphp3/lib \
            --with-exec-dir=/opt/bin \
            --enable-debug=yes \
            --enable-safe-mode=no \
            --enable-url-fopen-wrapper=yes \
            --enable-track-vars=yes \
            --prefix=/opt/GNUphp3
```

then the resulting PHP binary will be dynamically linked against libclntsh.so in /opt/oracle/oracle/lib. The build process as generated by configure will include this directory with an -L option, but not with an -R option (run-time library search path). This means that the resulting binary will not be able to find libclntsh.so by itself, but depends on an appropriate LD_LIBRARY_PATH set within the web server CGI environment - which usually is not there. To fix this, grab the Makefile generated by configure and duplicate the -L/opt/oracle/oracle/8.0.3/lib option as an -R/opt/oracle/oracle/8.0.3/lib option (your -L and -R strings will look slightly different depending on the values for your oracle version and ORACLE_HOME). The -R option will compile in the corrent search path for libclntsh and you won't have any LD_LIBRARY_PATH pains any more.

XXXVI. PDF functions

Table of Contents

¿©·¯ºÐÀÌ Thomas Merz°¡ Á¦°øÇÏ´Â PDF ¶óÀÌºê·¯¸®(http://www.ifconnection.de/~tm/¿¡¼- Ã£À» ¼ö ÀÖ´Ù.)¸¦ °¡Áö°í ÀÖ´Ù¸é ¿©·¯ºÐÀº PHP¿¡¼- pdf ÆÄÀÏÀ» ¸¸µé ¼ö ÀÖ´Â pdf ÇÔ¼öµéÀ» »ç¿ëÇÒ ¼ö ÀÖ´Ù. pdflib¿¡ ´ëÇÑ ÀÚ¼¼ÇÑ ¼³¸íÀº pdflibÀÇ ¼Ò½º¿Í ÇÔ²² ¹èÆ÷µÇ´Â °ÍÀ» ÀÐ¾î ºÁ³ª, http://www.ifconnection.de/~tm/software/pdflib/PDFlib - 0.6.pdf¿¡¼- ±¸ÇÏ¿© ÀÐ¾î¸¸¸é µÈ´Ù. ÀÌ PHP3 ¸Å´º¾óÀÌ ¿Ï¼ºµÇÁö ¾Ê´ÂÇÑ ¾Õ¿¡¼- ¼³¸íÇÑ pdflib ¹®¼-µéÀ» ¿ì¼± ÀûÀ¸·Î ºÁ¾Æ¾ß ÇÒ °ÍÀÌ´Ù. php3ÀÇ ¸ðµâ¿¡ ÀÖ´Â ÇÔ¼ö´Â pdflib ÇÔ¼ö¿Í °°Àº ÀÌ¸§À» »ç¿ëÇÑ´Ù. ¶ÇÇÑ ÆÄ¶ó¸ÞÅÍµµ µ¿ÀÏÇÏ´Ù. ¶ÇÇÑ ¿©·¯ºÐÀÌ ÀÌ ¸ðµâ» È¿À²ÀûÀ¸·Î »ç¿ëÇÏ·Á¸é ¿©·¯ºÐÀº ¾î´ÀÁ¤µµ pdfÀÇ °³³ä¿¡ ´ëÇØ ÀÌÇØÇÏ°í ÀÖ¾î¾ß ÇÑ´Ù. pdf ¸ðµâÀº *pdfdoc*¿Í *pdfinfo*¶ó´Â µÎ °¡Áö »õ·Î¿î °¯¼ö typeÀ» Á¦°øÇÑ´Ù.

PDF_get_info

PDF_get_info - - pdf ¹®¼-¿¡ ´ëÇÑ ±â°» Á¤º¸¸¦(info structure)¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

info pdf_get_info(string filename);

The **PDF_get_info()** function will return a default info structure for the pdf document. It can be filled with appropriate information like the author, subject etc.

Example 1. PDF_get_info

```
<?php $info = PDF_get_info();
PDF_set_info_creator($info, "Name of Author") ?>
```

See also **PDF_set_info_creator()**, **PDF_set_info_subject()**.

PDF_set_info_creator

PDF_set_info_creator - - info structureÀÇ creator ÇÊµå¸¦ ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_info_creator(info info, string creator);

The **PDF_set_info_creator()** function sets the creator of a pdf document. It has to be called after **PDF_get_info()** and before **PDF_open()**. Calling it after **PDF_open()** will have no effect on the document.

> **Note:** This function is not part of the pdf library.

See also **PDF_get_info()**, **PDF_set_info_subject()**.

PDF_set_info_title

PDF_set_info_title - - info structureÀÇ title ÇÊµå¸¦ ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_info_title(info info, string title);

The **PDF_set_info_title()** function sets the title of a pdf document. It has to be called after **PDF_get_info()** and before **PDF_open()**. Calling it after **PDF_open()** will have no effect on the document.

> **Note:** This function is not part of the pdf library.

See also **PDF_get_info()**, **PDF_set_info_xxxxx()**.

PDF_set_info_subject

PDF_set_info_subject - - info structureÀÇ subject ÇÊµå¸¦ ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_info_subject(info info, string subject);

The **PDF_set_info_subject()** function sets the subject of a pdf document. It has to be called after **PDF_get_info()** and before **PDF_open()**. Calling it after **PDF_open()** will have no effect on the document.

> **Note:** This function is not part of the pdf library.

See also **PDF_get_info()**, **PDF_set_info_xxxxx()**.

PDF_set_info_keywords

PDF_set_info_keywords - - info structureÀÇ keyword ÇÊµå¸¦ ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_info_keywords(info info, string keywords);

The **PDF_set_info_keywords()** function sets the keywords of a pdf document. It has to be called after **PDF_get_info()** and before **PDF_open()**. Calling it after **PDF_open()** will have no effect on the document.

> **Note:** This function is not part of the pdf library.

See also **PDF_get_info()**, **PDF_set_info_xxxxx()**.

PDF_set_info_author

PDF_set_info_author - - info structureÀÇ author ÇÊµå¸¦ ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_info_author(info info, string author);

The **PDF_set_info_author()** function sets the author of a pdf document. It has to be called after **PDF_get_info()** and before **PDF_open()**. Calling it after **PDF_open()** will have no effect on the document.

> **Note:** This function is not part of the pdf library.

See also **PDF_get_info()**, **PDF_set_info_xxxxx()**.

PDF_open

PDF_open - - »õ pdf ¹®¼-¸¦ ¿¬´Ù.

Description

`int pdf_open`(int descripterfile, int info);

The **PDF_set_info_author()** function opens a new pdf document. The corresponding file has to be opened with **fopen()** and the file descriptor passed as argument *file*. *info* is the an info structure that has to be created with **pdf_get_info()**.

> **Note:** The return value is needed as the first parameter in all other functions writing to the pdf document.

See also **fopen()**, **PDF_get_info()**.

PDF_close

PDF_close - - pdf ¹®¼-¸¦ ´Ý´Â´Ù.

Description

`void pdf_close`(int pdf document);

The **PDF_close()** function closes the pdf document *int*.

> **Note:** It will not close the file. You need to call an extra **fclose()** after **pdf_close()**.

See also **PDF_open()**, **fclose()**.

PDF_begin_page

PDF_begin_page - - »õ ÆäÀÌÁö¸¦ ½ÃÀÛÇÑ´Ù.

Description

`void pdf_begin_page`(int pdf document, double height, double width);

The **PDF_begin_page()** function starts a new page with height *height* and width *width*.

See also **PDF_end_page()**.

PDF_end_page

PDF_end_page - - ÆäÀÌÁö¸¦ ³¡³½´Ù.

Description

`void pdf_end_page`(int pdf document);

The **PDF_end_page()** function ends a page.

See also **PDF_end_page()**.

PDF_show

PDF_show - - ÇöÀç À§Ä¡¿¡ text¸¦ Ãâ·ÂÇÑ´Ù.

Description

void pdf_show(int pdf document, string text);

The **PDF_show()** function outputs the string in *text* at the current position.

See also **PDF_show_xy()**, **PDF_set_text_pos()**.

PDF_show_xy

PDF_show_xy - - ÁöÁ¤ÇÑ À§Ä¡¿¡ text¸¦ Ãâ·ÂÇÑ´Ù.

Description

void pdf_show_xy(int pdf document, string text, double x-koor, double y-koor);

The **PDF_show_xy()** function outputs the string in *text* at position with koordinates (*x-koor*, *y-koor*).

See also **PDF_show()**.

PDF_set_font

PDF_set_font - - »ç¿ëÇÒ font ¸ð¾ç°ú Å©±â¸¦ ¼±ÅÃÇÑ´Ù.

Description

void pdf_set_font(int pdf document, string font name, double size, string encoding);

The **PDF_set_font()** function sets the the current font face, font size and encoding. You will need to provide the Adobe Font Metrics (afm-files) for the font in the font path (default is ./fonts).

See also **PDF_info()**.

PDF_set_leading

PDF_set_leading - - ÁÙ °£°ÝÀ» ¼³Á¤ÇÑ´Ù.

Description

void pdf_set leading(int pdf document, double distance);

The **PDF_set_leading()** function sets the distance between text lines. This will be used if text is output by **PDF_continue_text()**.

See also **PDF_continue_text()**.

PDF_set_text_rendering

PDF_set_text_rendering - - text¸¦ Ãâ·ÂÇÒ ¹æ½ÄÀ» °áÁ¤ÇÑ´Ù.(Determines how text is rendered)

Description

void pdf_set_text_rendering(int pdf document, int mode);

The **PDF_set_text_rendering()** function determines how text is rendered. The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to cliping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to cliping path, 7=add it to clipping path.

PDF_set_horiz_scaling

PDF_set_horiz_scaling - - textÀÇ ¼öÆò È®´ë °ñÀ²À» Á¤ÇÑ´Ù.

Description

void pdf_set_horiz_scaling(int pdf document, double scale);

The **PDF_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

PDF_set_text_rise

PDF_set_text_rise - - text¸¦ riseÇÏ°Ô Ç¥½ÃµÇµµ·Ï ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_text_rise(int pdf document, double value);

The **PDF_set_text_rise()** function sets the text rising to *value* units.

PDF_set_text_matrix

PDF_set_text_matrix - - text¸¦ matrix ÇüÅÂ·Î ÁöÁ¤ÇÑ´Ù.

Description

void pdf_set_text_matrix(int pdf document, array matrix);

The **PDF_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font.

PDF_set_text_pos

PDF_set_text_pos - - textÀÇ À§Ä¡¸¦ ¼³Á¤ÇÑ´Ù.

Description

void pdf_set_text_pos(int pdf document, double x-koor, double y-koor);

The **PDF_set_text_pos()** function sets the position of text for the next **pdf_show()** function call.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_set_char_spacing

PDF_set_char_spacing - - ±ÛÀÚ »çÀÌÀÇ °£°ÝÀ» ÁöÁ¤ÇÑ´Ù.

Description

void pdf_set_char_spacing(int pdf document, double space);

The **PDF_set_char_spacing()** function sets the spacing between characters.

See also **PDF_set_word_spacing()**, **PDF_set_text_leading()**.

PDF_set_word_spacing

PDF_set_word_spacing - - ´Ü¾î »çÀÌÀÇ °£°ÝÀ» ÁöÁ¤ÇÑ´Ù.

Description

void pdf_set_word_spacing(int pdf document, double space);

The **PDF_set_word_spacing()** function sets the spacing between words.

See also **PDF_set_char_spacing()**, **PDF_set_text_leading()**.

PDF_continue_text

PDF_continue_text - - »õ ÁÙ¿¡ text¸¦ Ãâ·ÂÇÑ´Ù.

Description

void pdf_continue_text(int pdf document, string text);

The **PDF_continue_text()** function outputs the string in *text* in the next line.

See also **PDF_show_xy()**, **PDF_set_text_leading()**, **PDF_set_text_pos()**.

PDF_stringwidth

PDF_stringwidth - - ÇöÀç »ç¿ëÁßÀÎ fontÀÇ ³ÐÀÌ¸¦ ±¸Ñ´Ù.

Description

double pdf_stringwidth(int pdf document, string text);

The **PDF_stringwidth()** function returns the width of the string in *text*. It requires a font to be set before.

See also **PDF_set_font()**.

PDF_save

PDF_save - - ÇöÀç È¯°æÀ» ÀúÀåÇÑ´Ù.

Description

void pdf_save(int pdf document);

The **PDF_save()** function saves the current enviroment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects.

See also **PDF_restore()**.

PDF_restore

PDF_restore - - Àü¿¡ ÀúÀåÇØ µÎ¾ú´ø È¯°æÀ¸·Î ¹¹±ÍÑ´Ù.

Description

void pdf_restore(int pdf document);

The **PDF_restore()** function restores the enviroment saved with **PDF_save()**. It works like the postscript command grestore. Very useful if you want to translate or rotate an object without effecting other objects.
Example 1. PDF_get_info

```
<?php PDF_save($pdf);
// do all kinds of rotations, transformations, ...
PDF_restore($pdf) ?>
```

See also **PDF_save()**.

PDF_translate

PDF_translate - - koordinate systemÀÇ ±âÁØÀÌµÇ´Â ¿øÁ¡À» Á¤Ñ´Ù.

Description

void pdf_translate(int pdf document, double x-koor, double y-koor);

The **PDF_translate()** function set the origin of koordinate system to the point (*x-koor*, *y-koor*).

PDF_scale

PDF_scale - - scalingÀ» ¼³Á¤Ñ´Ù.

Description

void pdf_scale(int pdf document, double x-scale, double y-scale);

The **PDF_scale()** function set the scaling factor in both directions.

PDF_rotate

PDF_rotate - - È¸ÀüÀ²À» Á¤Ñ´Ù.

Description

void pdf_rotate(int pdf document, double angle);

The **PDF_rotate()** function set the rotation in degress to *angle*.

PDF_setflat

PDF_setflat - - flatness¸¦ Á¤ÇÑ´Ù.

Description

void pdf_setflat(int pdf document, double value);

The **PDF_setflat()** function set the flatness to a value between 0 and 100.

PDF_setlinejoin

PDF_setlinejoin - - linejoin parameter¸¦ Á¤ÇÑ´Ù.

Description

void pdf_setlinejoin(int pdf document, long value);

The **PDF_setlinejoin()** function set the linejoin parameter between a value of 0 and 2.

PDF_setlinecap

PDF_setlinecap - - linecap aparameter¸¦ Á¤ÇÑ´Ù.

Description

void pdf_setlinecap(int pdf document, int value);

The **PDF_setlinecap()** function set the linecap parameter between a value of 0 and 2.

PDF_setmiterlimit

PDF_setmiterlimit - - miter limitÀ» Á¤ÇÑ´Ù.

Description

void pdf_setmiterlimit(int pdf document, double value);

The **PDF_setmiterlimit()** function set the miter limit to a value greater of equal than 1.

PDF_setlinewidth

PDF_setlinewidth - - ÇÑ ¶óÀÎÀÇ ÆøÀ» Á¤ÇÑ´Ù.

Description

void pdf_setlinewidth(int pdf document, double width);

The **PDF_setlinewidth()** function set the line width to *width*.

PDF_setdash

PDF_setdash - - Á¡¼±ÀÇ ¹«´Ì¸¦ Á¤ÇÑ´Ù.

Description

void pdf_setdash(int pdf document, double white, double black);

The **PDF_setdash()** function set the dash pattern *white* white units and *black* black units. If both are 0 a solid line is set.

PDF_moveto

PDF_moveto - - ÇöÀç À§Ä¡¸¦ ¹Ù²Û´Ù.

Description

void pdf_moveto(int pdf document, double x-koor, double y-koor);

The **PDF_moveto()** function set the current point to the coordinates *x-koor* and *y-koor*.

PDF_curveto

PDF_curveto - - °î¼±À» ±×¸°´Ù.

Description

void pdf_curveto(int pdf document, double x1, double y1, double x2, double y2, double x3, double y3);

The **PDF_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

See also **PDF_moveto()**, **PDF_lineto()**.

PDF_lineto

PDF_lineto - - Á÷¼±À» ±×¸°´Ù.

Description

void pdf_lineto(int pdf document, double x-koor, double y-koor);

The **PDF_lineto()** function draws a line from the current point to the point with coordinates (*x-koor*, *y-koor*).

See also **PDF_moveto()**, **PDF_curveto()**.

PDF_circle

PDF_circle - - ¿øÀ» ±×¸°´Ù.

Description

void pdf_circle(int pdf document, double x-koor, double y-koor, double radius);

The **PDF_circle()** function draws a circle with center at point (*x-koor*, *y-koor*) and radius *radius*.

See also **PDF_arc()**.

PDF_arc

PDF_arc - - ¿øÈ£¸¦ ±×¸°´Ù.

Description

void pdf_arc(int pdf document, double x-koor, double y-koor, double radius, double start, double end);

The **PDF_arc()** function draws an arc with center at point (*x-koor*, *y-koor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

See also **PDF_circle()**.

PDF_rect

PDF_rect - - »ç°¢ÇüÀ» ±×¸°´Ù.

Description

void pdf_rect(int pdf document, double x-koor, double y-koor, double width, double height);

The **PDF_rect()** function draws a rectangle with its lower left corner at point (*x-koor*, *y-koor*). This width is set to *widgth*. This height is set to *height*.

PDF_closepath

PDF_closepath - - path¸¦ ´Ý´Â´Ù.

Description

void pdf_closepath(int pdf document);

The **PDF_closepath()** function closes the current path.

PDF_stroke

PDF_stroke - - path¸¦ µû¶ó ¼±À» ±×¸°´Ù.

Description

void pdf_stroke(int pdf document);

The **PDF_stroke()** function draws a line along current path.

See also **PDF_closepath()**, **PDF_closepath_stroke()**.

PDF_closepath_stroke

PDF_closepath_stroke - - path¸¦ µû¶ó ¼±À» ±×¸®°í path¸¦ ´Ý´Â´Ù.

Description

void pdf_closepath_stroke(int pdf document);

The **PDF_closepath_stroke()** function is a combination of **PDF_closepath()** and **PDF_stroke()**. Than clears the path.

See also **PDF_closepath()**, **PDF_stroke()**.

PDF_fill

PDF_fill - - ÇöÀç path¸¦ Ä¥ÇÑ´Ù.

Description

void pdf_fill(int pdf document);

The **PDF_fill()** function fills the interior of the current path with the current fill color.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_fill_stroke

PDF_fill_stroke - - ÇöÀç pathÀÇ ¸¶ÂîÊÀ» Ä¥ÇÏ°í, ÇöÀç path¸¦ µû¶ó ±×¸°´Ù.

Description

void pdf_fill_stroke(int pdf document);

The **PDF_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_closepath_fill_stroke

PDF_closepath_fill_stroke - - ÇöÀç pathÀÇ ¸¶ÂîÊÀ» Ä¥ÇÏ°í, ÇöÀç path¸¦ µû¶ó ±×¸° ÈÄ, path¸¦ ´Ý´Â´Ù.

Description

void pdf_closepath_fill_stroke(int pdf document);

The **PDF_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_endpath

PDF_endpath - - ÇöÀç path¸¦ Á¾·áÇÑ´Ù.

Description

void pdf_endpath(int pdf document);

The **PDF_endpath()** function ends the current path but does not close it.

See also **PDF_closepath()**.

PDF_clip

PDF_clip - - ÇöÀç path¸¦ clipÇÑ´Ù.

Description

void pdf_clip(int pdf document);

The **PDF_clip()** function clips all drawing to the current path.

PDF_setgray_fill

PDF_setgray_fill - - Ä¥ÇÏ´Â »öÀ» È¸»öÅæÀ¸·Î ¼³Á¤ÇÑ´Ù.

Description

void pdf_setgray_fill(int pdf document, double value);

The **PDF_setgray_fill()** function sets the current gray value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setgray_stroke

PDF_setgray_stroke - - ±×¸®´Â »öÀ» È¸»öÅæÀ¸·Î ¼³Á¤ÇÑ´Ù.

Description

void pdf_setgray_stroke(int pdf document, double gray value);

The **PDF_setgray_stroke()** function sets the current drawing color to the given gray value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setgray

PDF_setgray - - Ä¥ÇÏ°í ±×¸®´Â »öÀ» È¸»öÅæÀ¸·Î ¼³Á¤ÇÑ´Ù.

Description

void pdf_setgray(int pdf document, double gray value);

The **PDF_setgray_stroke()** function sets the current drawing and filling color to the given gray value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_setrgbcolor_fill

PDF_setrgbcolor_fill - - Ä¥ÇÏ´Â »öÀ» rgb color·Î ¼³Á¤ÇÑ´Ù.

Description

void pdf_setrgbcolor_fill(int pdf document, double red value, double green value, double blue value);

The **PDF_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setrgbcolor_stroke

PDF_setrgbcolor_stroke - - ±×¸®´Â »öÀ» rgb color·Î ¼³Á¤ÇÑ´Ù.

Description

void pdf_setrgbcolor_stroke(int pdf document, double red value, double green value, double blue value);

The **PDF_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setrgbcolor

PDF_setrgbcolor - - Ä¥ÇÏ°í ±×¸®´Â »öÀ» rgb color·Î ¼³Á¤ÇÑ´Ù.

Description

void pdf_setrgbcolor(int pdf document, double red value, double green value, double blue value);

The **PDF_setrgbcolor_stroke()** function sets the current drawing and filling color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_add_outline

PDF_add_outline - - ÇöÀç ÆäÀÌÁö¿¡ bookmark¸¦ Ãß°¡ÇÑ´Ù.

Description

void pdf_add_outline(int pdf document, string text);

The **PDF_add_outline()** function adds a bookmark with text *text* that points to the current page.

PDF_set_transition

PDF_set_transition - - ÆäÀÌÁö°£ÀÇ transitionÀ» ÁöÁ¤ÇÑ´Ù.

Description

void pdf_set_transition(int pdf document, int transition);

The **PDF_set_transition()** function set the transition between following pages. The value of *transition* can be 0 for none, 1 for two lines sweeping across the screen reveal the page, 2 for multiple lines sweeping across the screen reveal the page, 3 for a box reveals the page, 4 for a single line sweeping across the screen reveals the page, 5 for the old page dissolves to reveal the page, 6 for the dissolve effect moves from one screen edge to another, 7 for the old page is simply replaced by the new page (default)

PDF_set_duration

PDF_set_duration - - page°£ÀÇ durationÀ» ÁöÁ¤ÇÑ´Ù.

Description

void pdf_set_duration(int pdf document, double duration);

The **PDF_set_duration()** function set the duration between following pages in seconds.

PDF_open_jpeg

PDF_open_jpeg - - JPEG image¸¦ ¿¬´Ù.

Description

int pdf_open_jpeg(int pdf document, string file name);

The **PDF_open_jpeg()** function opens an image stored in the file with the name *file name*. The format of the image has to be jpeg.

See also **PDF_close_image()**,

PDF_close_image

PDF_close_image - - image¸¦ ´Ý´Â´Ù.

Description

void pdf_close_image(int image);

The **PDF_close_image()** function closes an image which has been opened with any of the **PDF_open_xxx()** functions.

See also **PDF_open_jpeg()**,

PDF_place_image

PDF_place_image - - image¸¦ page¿¡ À§Ä¡½ÃÅ²´Ù.

Description

void pdf_place_image(int pdf document, int image, double x-koor, double y-koor, double skale);

The **PDF_place_image()** function places an image on the page at postion (*x-koor*, *x-koor*). The image can be skaled at the same time.

PDF_put_image

PDF_put_image - - ³ªÁß¿¡ ¾²±â À§ÇØ image¸¦ PDF¿¡ ÀúÀåÇÑ´Ù.

Description

void pdf_put_image(int pdf document, int image);

The PDF_put_image function places an image in the PDF file without showing it. The stored image can be displayed with the **PDF_execute_image()** function. This is useful when using the same image multiple times.

PDF_execute_image

PDF_execute_image - - ÀúÀåµÈ image¸¦ page¿¡ À§Ä¡½ÃÅ²´Ù.

Description

void pdf_execute_image(int pdf document, int image, double x-coor, double y-coor, double scale);

The PDF_execute_image function displays an image that has been put in the PDF file with the **PDF_put_image()** function on the current page at the given coordinates.

The image can be scaled while displaying it. A scale of 1.0 will show the image in the original size.

XXXVII. PostgreSQL functions

Table of Contents

UC Berkeley Computer Science Department¿¡¼- Ã³À½ °³¹ßµÈ Postgres´Â ÀÌÁ¦´Â ÀÏºÎ »ó¿ë database¿¡¼- °¡´ÉÇÑ object-relational °³³äÀÇ ¸ðµ¨À» °³Ã´ÇÏ¿´´Ù. ÀÌ°ÍÀº SQL92/SQL3 ¾ð¾î¸¦ Áö¿øÇÏ°í, transaction integrity¿Í type È® Àå°¡´É¼º À» Á¦°øÇÑ´Ù. PostgreSQLÀº public-domainÀÌ°í, original Berkeley codeÀÇ ¹ßÀüµÈ ¼Ò½º¸¦ Á¦°øÇÑ´Ù.

PostgreSQLÀº ¹«·á·Î »ç¿ë°¡´ÉÇÏ´Ù. ÇöÀç ¹öÁ¯Àº 6.3.3±îÁö ³ª¿ÍÀÖ°í, www.postgreSQL.org¿¡¼- Ã£À» ¼ö ÀÖ´Ù.

version 6.3 (03/02/1998)ÀÌÈÄ·Î PostgreSQLÀº unix domain socketÀ» »ç¿ëÇÏ°í, ÀÌ ±â´ÉÀ» À§ÇØ tableÀÌ ÁÖ¾îÁø´Ù. ÀÌ ¼ÒÄÏÀº `/tmp/.s.PGSQL.5432`¿¡¼- Ã£À» ¼ö ÀÖ´Ù. ÀÌ ¿É¼ÇÀº **postmaster**¿¡ '-i' flag¸¦ ÁÖ¾î¼- enable½ÅÅ³ ¼ö ÀÖ´Â¥, ÀÌ°ÍÀº "TCP/IP sockets»Ó ¾Æ´Ï¶ó Unix domain socketÀÇ ¿äÃ»µµ ¹Þ¾Æµé¿©¶ó."´Â ÀÇ¹ÌÀÌ´Ù.

Table 1. Postmaster and PHP

Postmaster	PHP	Status
postmaster &	pg_connect("", "", "", "", "dbname");	OK
postmaster -i &	pg_connect("", "", "", "", "dbname");	OK
postmaster &	pg_connect("localhost", "", "", "", "dbname");	PostgreSQL server¿¡ ¿¬°áÇÒ ¼ö ¾øÀ½: 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

large object (lo) interface¸¦ »ç¿ëÇÏ·Á¸é transaction blockÀ¸·Î µÑ·¯½Î´Â °ÍÀÌ ÇÊ¿äÇÏ´Ù. transaction blockÀº **begin**À¸·Î ½ÃÀÛÇÏ°í, transactionÀÌÈ®½ÇÇÏ´Ù¸é **commit**°ú **end**·Î ³¡³½´Ù. ¸¸¾à transactionÀÌ ¿À·ù¸¦ Å³ª¾ª ½ÇÆÐÇÏ¸é transaction blockÀº **abort**¿Í **rollback**À¸·Î ³¡³»¾ß ÇÒ °ÍÀÌ´Ù.

Example 1. Using Large Objects

```
<?php
$database = pg_Connect ("", "", "", "", "jacarta");
pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
pg_exec ($database, "commit")
pg_exec ($database, "end")
?>
```

---

pg_Close

pg_Close - - PostgreSQL connectionÀ» ´Ý´Â´Ù.

Description

void pg_close(int connection);

Returns false if connection is not a valid connection index, true otherwise. Closes down the connection to a PostgreSQL database associated with the given connection index.

pg_cmdTuples

pg_cmdTuples - - ÇØ´çÇÏ´Â tupleÀÇ °³¼ö¸¦ ±¸Ñ´Ù.

Description

int pg_cmdtuples(int result_id);

pg_cmdTuples() returns the number of tuples (instances) affected by INSERT, UPDATE, and DELETE queries. If no tuple is affected the function will return 0.
Example 1. pg_cmdtuples

```
<?php
$result = pg_exec($conn, "INSERT INTO verlag VALUES ('Autor')");
$cmdtuples = pg_cmdtuples($result);
echo $cmdtuples . " <- cmdtuples affected.";
?>
```

pg_Connect

pg_Connect - - connectionÀ» ¿¬´Ù.

Description

int pg_connect(string host, string port, string options, string tty, string dbname);

Returns a connection index on success, or false if the connection could not be made. Opens a connection to a PostgreSQL database. Each of the arguments should be a quoted string, including the port number. The options and tty arguments are optional and can be left out. This function returns a connection index that is needed by other PostgreSQL functions. You can have multiple connections open at once.

A connection can also established with the following command: **$conn = pg_connect("dbname=marliese port=5432");** Other parameters besides *dbname* and *port* are *host*, *tty* and *options*.

See also **pg_pConnect()**.

pg_DBname

pg_DBname - - database ÀÌ¸§

Description

string pg_dbname(int connection);

Returns the name of the database that the given PostgreSQL connection index is connected to, or false if connection is not a valid connection index.

pg_ErrorMessage

pg_ErrorMessage - - ¿¡·¯ ¸Þ¼¼Áö

Description

string pg_errormessage(int connection);

Returns a string containing the error message, false on failure. Details about the error probably cannot be retrieved using the pg_errormessage() function if an error occured on the last database action for which a valid connection exists, this function will return a string containing the error message generated by the backend server.

pg_Exec

pg_Exec - - ÁúÀÇ¸¦ ½ÇÇàÇÑ´Ù.

Description

int pg_exec(int connection, string query);

Returns a result index if query could be executed, false on failure or if connection is not a valid connection index. Details about the error can be retrieved using the **pg_ErrorMessage()** function if connection is valid. Sends an SQL statement to the PostgreSQL database specified by the connection index. The connection must be a valid index that was returned by **pg_Connect()**. The return value of this function is an index to be used to access the results from other PostgreSQL functions.

> **NOTE:** PHP2 returned 1 if the query was not expected to return data (inserts or updates, for example) and greater than 1 even on selects that did not return anything. No such assumption can be made in PHP3.

pg_Fetch_Array

pg_Fetch_Array - - row¸¦ ¹è¿-·Î °¡Á®¿Â´Ù.

Description

`array pg_fetch_array`(int result, int row);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

pg_fetch_array() is an extended version of **pg_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **pg_fetch_array()** is NOT significantly slower than using **pg_fetch_row()**, while it provides a significant added value.

For further details, also see **pg_fetch_row()**
Example 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}
$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}
$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n";
$arr = pg_fetch_array ($result, 1);
echo $arr["author"] . " <- array\n";
?>
```

pg_Fetch_Object

pg_Fetch_Object - - row¸¦ object·Î °¡Á®¿Â´Ù.

Description

`object pg_fetch_object`(int result, int row);

Returns: An object with properties that correspond to the fetched row, or false if there are no more rows.

pg_fetch_object() is similar to **pg_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **pg_fetch_array()**, and almost as quick as **pg_fetch_row()** (the difference is insignificant).

See also: **pg_fetch_array()** and **pg_fetch_row()**.

Example 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <? echo $database ?></H1> <?
    exit;
endif;
$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>
<PRE><?
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");
$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)):
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "----------\n"; ?>
</PRE>
```

pg_Fetch_Row

pg_Fetch_Row - - row¸¦ ¹è¿(enumerated array)·Î °¡Á®¿Â´Ù.

Description

array pg_fetch_row(int result, int row);

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

pg_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **pg_fetch_row()** would return the next row in the result set, or false if there are no more rows.

See also: **pg_fetch_array()**, **pg_fetch_object()**, **pg_result()**.

Example 1. Postgres fetch row

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}
$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}
$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";
$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";
$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n";
?>
```

pg_FieldIsNull

pg_FieldIsNull - - field°¡ NULLÀÌ°¡ °Ë»çÇÑ´Ù.

Description

int pg_fieldisnull(int result_id, int row, mixed field);

Test if a field is NULL or not. Returns 0 if the field in the given row is not NULL. Returns 1 if the field in the given row is NULL. Field can be specified as number or fieldname. Row numbering starts at 0.

pg_FieldName

pg_FieldName - - fieldÀÇ ÀÌ¸§À» ¹ÝÈ¯ÇÑ´Ù.

Description

string pg_fieldname(int result_id, int field_number);

pg_FieldName() will return the name of the field occupying the given column number in the given PostgreSQL result identifier. Field numbering starts from 0.

pg_FieldNum

pg_FieldNum - - columnÀÇ °³¼ö¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

string pg_fieldnum(int result_id, int field_name);

pg_FieldNum() will return the number of the column slot that corresponds to the named field in the given PosgreSQL result identifier. Field numbering starts at 0. This function will return -1 on error.

pg_FieldPrtLen

pg_FieldPrtLen - - ÇÁ¸°µåµÉ ±æÀÌ(¹®ÀÚ °³¼ö)¸¦ ±¸ÇÑ´Ù.

Description

int pg_fieldprtlen(int result_id, int row_number, string field_name);

pg_FieldPrtLen() will return the actual printed length (number of characters) of a specific value in a PostgreSQL result. Row numbering starts at 0. This function will return -1 on an error.

pg_FieldSize

pg_FieldSize - - ÁöÁ¤µÈ fieldÀÇ ÀúÀåÀ» À§ÇØ ÇÒ´çµÈ °ø°£(internal storage)ÀÇ Å©±â

Description

int pg_fieldsize(int result_id, string field_name);

pg_FieldSize() will return the internal storage size (in bytes) of the named field in the given PostgreSQL result. A field size of -1 indicates a variable length field. This function will return false on error.

pg_FieldType

pg_FieldType - - ÇØ´ç ÇÊµå ¹øÈ£¿¡ ´ëÀÀÇÏ´Â typeÀÌ¸§À» ¹ÝÈ¯ÇÑ´Ù.

Description

int pg_fieldtype(int result_id, int field_number);

pg_FieldType() will return a string containing the type name of the given field in the given PostgreSQL result identifier. Field numbering starts at 0.

pg_FreeResult

pg_FreeResult - - result memory¸¦ ÇØÁ¦ÇÑ´Ù.

Description

int pg_freeresult(int result_id);

pg_FreeResult() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_FreeResult()** with the result identifier as an argument and the associated result memory will be freed.

pg_GetLastOid

pg_GetLastOid - - ÃÕ±ÙÀÇ object identifier¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int pg_getlastoid(int result_id);

pg_GetLastOid() can be used to retrieve the Oid assigned to an inserted tuple if the result identifier is used from the last command sent via **pg_Exec()** and was an SQL INSERT. This function will return a positive integer if there was a valid Oid. It will return -1 if an error occured or the last command sent via **pg_Exec()** was not an INSERT.

pg_Host

pg_Host - - host ÀÌ¸§À» ¹ÝÈ¯ÇÑ´Ù.

Description

string pg_host(int connection_id);

pg_Host() will return the host name of the given PostgreSQL connection identifier is connected to.

pg_loclose

pg_loclose - - large object¸¦ ´Ý´Â´Ù.

Description

void pg_loclose(int fd);

pg_loclose() closes an Inversion Large Object. *fd* is a file descriptor for the large object from **pg_loopen()**.

pg_locreate

pg_locreate - - large object¸¦ »ý¼ºÇÑ´Ù.

Description

int pg_locreate(int conn);

pg_locreate() creates an Inversion Large Object and returns the oid of the large object. *conn* specifies a valid database connection. PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above).

pg_loopen

pg_loopen - - large object¸¦ ¿¬´Ù.

Description

int pg_loopen(int conn, int objoid, string mode);

pg_loopen() open an Inversion Large Object and returns file descriptor of the large object. The file descriptor encapsulates information about the connection. Do not close the connection before closing the large object file descriptor. *objoid* specifies a valid large object oid and *mode* can be either "r", "w", or "rw".

pg_loread

pg_loread - - large object¸¦ ÀÐ´Â´Ù.

Description

string pg_loread(int fd, int len);

pg_loread() reads at most *len* bytes from a large object and returns it as a string. *fd* specifies a valid large object file descriptor and *len* specifies the maximum allowable size of the large object segment.

pg_loreadall

pg_loreadall - - large object ÀüÃ¼¸¦ ÀÐ´Â´Ù.

Description

void pg_loreadall(int fd);

pg_loreadall() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound.

pg_lounlink

pg_lounlink - - large object¸¦ Áö¿î´Ù.

Description

void pg_lounlink(int conn, int lobjid);

pg_lounlink() deletes a large object with the *lobjid* identifier for that large object.

pg_lowrite

pg_lowrite - - large object¿¡ ¾´´Ù.

Description

int pg_lowrite(int fd, string buf);

pg_lowrite() writes at most to a large object from a variable *buf* and returns the number of bytes actually written, or false in the case of an error. *fd* is a file descriptor for the large object from **pg_loopen()**.

pg_NumFields

pg_NumFields - - fieldÀÇ °³¼ö¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int pg_numfields(int result_id);

pg_NumFields() will return the number of fields (columns) in a PostgreSQL result. The argument is a valid result identifier returned by **pg_Exec()**. This function will return -1 on error.

pg_NumRows

pg_NumRows - - rowÀÇ °³¼ö¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int pg_numrows(int result_id);

pg_NumRows() will return the number of rows in a PostgreSQL result. The argument is a valid result identifier returned by **pg_Exec()**. This function will return -1 on error.

pg_Options

pg_Options - - option¹éÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

string pg_options(int connection_id);

pg_Options() will return a string containing the options specified on the given PostgreSQL connection identifier.

pg_pConnect

pg_pConnect - - ¿µ±¸ÀûÀÎ database connectionÀ» ¸¸µç´Ù.

Description

int pg_pconnect(string host, string port, string options, string tty, string dbname);

Returns a connection index on success, or false if the connection could not be made. Opens a persistent connection to a PostgreSQL database. Each of the arguments should be a quoted string, including the port number. The options and tty arguments are optional and can be left out. This function returns a connection index that is needed by other PostgreSQL functions. You can have multiple persistent connections open at once. See also **pg_Connect()**.

A connection can also established with the following command: **$conn = pg_pconnect("dbname=marliese port=5432");** Other parameters besides *dbname* and *port* are *host*, *tty* and *options*.

pg_Port

pg_Port - - port ¹øÈ£¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int pg_port(int connection_id);

pg_Port() will return the port number that the given PostgreSQL connection identifier is connected to.

pg_Result

pg_Result - - result identifier·Î ºÎÅÍ °ªµéÀ» ±¸Ñ´Ù.

Description

mixed pg_result(int result_id, int row_number, mixed fieldname);

pg_Result() will return values from a result identifier produced by **pg_Exec()**. The *row_number* and *fieldname* sepcify what cell in the table of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of integer, boolean and oid types are returned as integer values. All forms of float, and real types are returned as double values. All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_tty

pg_tty - - tty ÀÌ¸§À» ¹ÝÈ¯ÇÑ´Ù.

Description

string pg_tty(int connection_id);

pg_tty() will return the tty name that server side debugging output is sent to on the given PostgreSQL connection identifier.

XXXVIII. Regular expression functions

Table of Contents

ereg

ereg - - regular expression ÀÌ ÁÖ¾îÁø ÆÐÅÏ¿¡ ¸Â´ÂÁö °Ë»çÇÑ´Ù.

Description

int ereg(string pattern, string string, array [regs]);

Searchs *string* for matches to the regular expression given in *pattern*.

If matches are found for parenthesized substrings of *pattern* and the function is called with the third argument *regs*, the matches will be stored in the elements of *regs*. $regs[1] will contain the substring which starts at the first left parenthesis; $regs[2] will contain the substring starting at the second, and so on. $regs[0] will contain a copy of *string*.

Searching is case sensitive.

Returns true if a match for pattern was found in string, or false if no matches were found or an error occurred.

The following code snippet takes a date in ISO format (YYYY - MM - DD) and prints it in DD.MM.YYYY format:
Example 1. ereg() example

```
if ( ereg( "([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs ) ) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
```

See also **eregi()**, **ereg_replace()**, and **eregi_replace()**.

ereg_replace

ereg_replace - - regular expressionÀÇ ¹®ÀåÀ» ´ëÄ¡ÇÑ´Ù.

Description

string ereg_replace(string pattern, string replacement, string string);

This function scans *string* for matches to *pattern*, then replaces the matched text with *replacement*.

If *pattern* contains parenthesized substrings, *replacement* may contain substrings of the form *digit*, which will be replaced by the text matching the digit'th parenthesized substring; \\0 will produce the entire contents of string. Up to nine substrings may be used. Parentheses may be nested, in which case they are counted by the opening parenthesis. For example, the following code snippet prints "This was a test" three times:
Example 1. ereg_replace() example

```
$string = "This is a test";
echo ereg_replace( " is", " was", $string );
echo ereg_replace( "( )is", "\\1was", $string );
echo ereg_replace( "(( )is)", "\\2was", $string );
```

See also **ereg()**, **eregi()**, and **eregi_replace()**.

eregi

eregi - - regular expression ÀÌ ÁÖ¾îÁø ÆÐÅÏ¿¡ ¸Â´ÂÁö ´ë,¼Ò¹®ÀÚ °¡¸®Áö ¾Ê°í °Ë»çÇÑ´Ù.

Description

int eregi(string pattern, string string, array [regs]);

This function is identical to **ereg()** save that this ignores case distinction when matching alphabetic characters.

See also **ereg()**, **ereg_replace()**, and **eregi_replace()**.

eregi_replace

eregi_replace - - regular expressionÀÇ ¹®ÀåÀ» ´ë,¼Ò¹®ÀÚ °¡¸®Áö ¾Ê°í ´ëÄ¡ÇÑ´Ù.

Description

string eregi_replace(string pattern, string replacement, string string);

This function is identical to **ereg_replace()** save that this ignores case distinction when matching alphabetic characters.

See also **ereg()**, **eregi()**, and **ereg_replace()**.

split

split - - ¹®ÀÚ¿-À» regular expression¿¡ ÀÇÇØ ³ª´«´Ù.

Description

array split(string pattern, string string, int [limit]);

Returns an array of strings, each of which is a substring of string formed by splitting it on boundaries formed by *pattern*. If an error occurs, returns false.

To get the first five fields from a line from /etc/passwd:
Example 1. split() example

```
$passwd_list = split( ":", $passwd_line, 5 );
```

Note that *pattern* is case-sensitive.

See also **explode()** and **implode()**.

sql_regcase

sql_regcase - - ´ë,¼Ò¹®ÀÚ °¡¸®Áö ¾Ê´Â °Ë»ç¸¦ À§ÇÑ regular expressionÀ» ¸¸µç´Ù.

Description

string sql_regcase(string string);

Returns a valid regular expression which will match *string*, ignoring case. This expression is *string* with each character converted to a bracket expression; this bracket expression contains that character's uppercase and lowercase form if applicable, otherwise it contains the original character twice.
Example 1. sql_regcase() example

```
echo sql_regcase( "Foo bar" );
```

prints

```
[Ff][Oo][Oo][ ][Bb][Aa][Rr]
```

.

This can be used to achieve case insensitive pattern matching in products which support only case sensitive regular expressions.

XXXIX. Semaphore and Shared Memory Functions

Table of Contents

ÀÌ ¸ðµâÀº System V ¼¼¸¶Æ÷¾î¸¦ »ç¿ëÇÏ´Â ¼¼¸¶Æ÷¾î ÇÔ¼ö¸¦ Á¦°øÇÑ´Ù. ¼¼¸¶Æ÷¾î´Â °¢ÅÍ ÇöÀç ÄÄÇ»ÅÍÀÇ ÀÚ¿ø(resources)¿¡ ´ëÇØ ¹èÅ¸ÀûÀ¸·Î Á¢±ÙÇÏ°Å³ª, ÇÑ ÀÚ¿ø¿¡ µ¿½Ã¿¡ »ç¿ë °¡´ÉÇÑ ÇÁ·Î¼¼½º¸¦ ¼ö¸¦ Á¦ÇÑ ÇÒ ¶§ »ç¿ëÇÑ´Ù.

¶ÇÇÑ ÀÌ ¸ðµâÀº System V °øÀ¯ ¸Þ¸ð¸®¸¦ »ç¿ëÇÏ´Â °øÀ¯ ¸Þ¸ð¸® ÇÔ¼ö¸¦ Á¦°øÇÑ´Ù. °øÀ¯ ¸Þ¸ð¸®(Shared memory)´Â Àü¿ª ¹¯¼ö¿¡ Á¢±ÙÇÏ±â¸§ÇØ »ç¿ëµÈ´Ù. ´Ù¸¥Á¡ÀÌ httpd-daemonÀÌ³ª ½ÉÁö¾î PerlÀÌ³ª C·Î ÀÛ¼º¸È ´Ù¸¥ ÇÁ·Î±×·¥¿¡¼-µµ ÀÌ ¹¯¼ö

¿¡ Á¢±ÙÇÒ ¼ö ÀÖÀ¸¸ ¹Ç·Î ±¤¹üÀ§ÇÑ µ¥ÀÌÅÍ ±³È¯ÀÌ °¡´ÉÇÏ´Ù. ÁÖÀÇÇÒ Á¡À° °øÀ¯ ¸Þ¸ð¸®´Â µ¿½Ã¿¡ µÎ±°µ¥¼- Á¢±ÙÇÏ´Â °Í¿¡ ´ëÇØ Àý´ë·Î ¾ÈÀüÇÏÁö ¾Ê´Ù. µû¶ó¼- ÀÌ¸¦ ¹æÁöÇÏ±â¸§ÇØ ¼¼¸¶Æ÷¾î¸¦ »ç¿ëÇÏ¿© µ¿±âÈ-(synchronize)ÇÏ¿©¾ß ÇÑ´Ù.

Table 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

sem_get

sem_get - - semaphore id¸¦ ¾ò´Â´Ù.

Description

`int sem_get`(int key, int [max_acquire], int [perm]);

Returns: A positive semaphore identifier on success, or false on error.

sem_get() returns an id that can be used to access the System V semaphore with the given key. The semaphore is created if necessary using the permission bits specified in perm (defaults to 0666). The number of processes that can acquire the semaphore simultaneously is set to max_acquire (defaults to 1). Actually this value is set only if the process finds it is the only process currently attached to the semaphore.

A second call to **sem_get()** for the same key will return a different semaphore identifier, but both identifiers access the same underlying semaphore.

See also: **sem_acquire()** and **sem_release()**.

sem_acquire

sem_acquire - - semaphore¸¦ ¾ò´Â´Ù.

Description

`int sem_acquire`(int sem_identifier);

Returns: true on success, false on error

sem_acquire() blocks (if necessary) until the semaphore can be acquired. A process attempting to acquire a semaphore which it has already acquired will block forever if acquiring the semaphore would cause its max_acquire value to be exceeded.

After processing a request, any semaphores acquired by the process but not explicitly released will be released automatically and a warning will be generated.

See also: **sem_get()** and **sem_release()**.

sem_release

sem_release - - semaphore¸¦ Ç®¾îÁØ´Ù.

Description

`int sem_release`(int sem_identifier);

Returns: true on success, false on error

sem_release() releases the semaphore if it is currently acquired by the calling process, otherwise a warning is generated.

After releasing the semaphore, **sem_acquire()** may be called to re - acquire it.

See also: **sem_get()** and **sem_acquire()**.

shm_attach

shm_attach - - °øÀ¯ ¸Þ¸ð¸®ÀÇ segment¸¦ ¸¸µé°Å³ª ¿¬´Ù.

Description

int shm_attach(int key, int [memsize], int [perm]);

shm_attach() returns an id that that can be used to access the System V shared memory with the given key, the first call creates the shared memory segment with mem_size (default: sysvshm.init_mem in php3.ini, otherwise 10000 bytes) and the optional perm - bits (default: 666).

A second call to **shm_attach()** for the same *key* will return a different shared memory identifier, but both identifiers access the same underlying shared memory. *memsize* and *perm* will be ignored.

shm_detach

shm_detach - - Disconnects from shared memory segment

Description

int shm_detach(int shm_identifier);

shm_detach() disconnects from the shared memory given by the *shm_identifier* created by **shm_attach()**. Remember, that shared memory still exist in the Unix system and the data is still present.

shm_get_var

shm_get_var - - Returns a variable from shared memory

Description

mixed shm_get_var(int id, int variable_key);

shm_get_var() returns the variable with a given *variable_key*. The variable is still present in the shared memory.

shm_put_var

shm_put_var - - Inserts or updates a variable in shared memory

Description

int shm_put_var(int shm_identifier, int variable_key, mixed variable);

Inserts or updates a *variable* with a given *variable_key*. All variable - types (double, int, string, array) are supported.

shm_remove

shm_remove - - Removes shared memory from Unix systems

Description

int shm_remove(int shm_identifier);

Removes shared memory from Unix systems. All data will be destroyed.

shm_remove_var

shm_remove_var - - Removes a variable from shared memory

Description

int shm_remove_var(int id, int variable_key);

Removes a variable with a given *variable_key* and frees the occupied memory.

XL. Solid Functions

Table of Contents

Solid ÇÔ¼öµéÀº ÁÁÀº Æò°¡¸¦ ¹ÞÁö ¸øÇÏ°í ÀÖ´Ù. ¾Æ¸¶µµ Unified ODBC functionsÀ» ´ë½Å »ç¿ëÇÏ´Â °ÍÀÌ ÁÁÀ» °ÍÀÌ´Ù.

solid_close

solid_close - - Solid connectionÀ» ´Ý´Â´Ù.

Description

See **odbc_close()**.

solid_connect

solid_connect - - Solid data source¿¡ ¿¬°áÇÑ´Ù.

Description

See **odbc_connect()**.

solid_exec

solid_exec - - Solid ÁúÀÇ¸¦ ½ÇÇàÇÑ´Ù.

Description

See **odbc_exec()**.

solid_fetchrow

solid_fetchrow - - Solid ÁúÀÇ·Î°ÎÅÍ µ¥ÀÌÅ¸ÀÇ row¸¦ °¡Á®¿Â´Ù.

Descriptio

See **odbc_fetch_row()**

solid_fieldname

solid_fieldname - - Solid ÁúÀÇ·Î°ÎÅÍ columnÀÇ ÀÌ¸§À» ±¸Ñ´Ù.

Description

See **odbc_field_name()**.

solid_fieldnum

solid_fieldnum - - Solid ÁúÀÇ·Î°ÎÅÍ columnÀÇ index¸¦ ±¸Ñ´Ù.

Description

See **odbc_field_num()**.

solid_freeresult

solid_freeresult - - Solid ÁúÀÇ·Î°ÎÅÍ result memory¸¦ ÇØÁ¦ÇÑ´Ù.

Description

See **odbc_free_result()**.

solid_numfields

solid_numfields - - Solid resultÀÇ field °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_num_fields()**.

solid_numrows

solid_numrows - - Solid resultÀÇ row °³¼ö¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_num_rows()**.

solid_result

solid_result - - Solid result·ÎºÎÅÍ data¸¦ ±¸ÇÑ´Ù.

Description

See **odbc_result()**.

XLI. SNMP Functions

Table of Contents

ÀÌ SNMP ÇÔ¼öµéÀ» »ç¿ëÇÏ·Á¸é ucd-snmp package¸¦ ¼³Ä¡ÇÏ¿©¾ß ÇÑ´Ù. ÃÖ½Å¹öÀüÀÌ ÀÖ´Â °÷¿¡ ´ëÇØ¼´Â PHP FAQ¸¦ ÂüÁ¶ÇÏ¸µ· ÏÇÑ´Ù. Windows¿¡¼ ÀÌ ÇÔ¼öµéÀº NTÈ¯°æ¿¡¼¸¸ µ¿ÀÛÇÏ¸ç Win95/98¿¡¼´Â µ¿ÀÛÇÏÁö ¾Ê´Â´Ù.

snmpget

snmpget - - SNMP object¸¦ °¡Á®¿Â´Ù.

Description

int snmpget(string hostname, string community, string object_id);

Returns SNMP object value on success and false on error.

The snmpget() function is used to read the value of an SNMP object specified by the object_id. SNMP agent is specified by the hostname and the read community is specified by the community parameter.

snmpget("127.0.0.1", "public", "system.SysContact.0")

snmpwalk

snmpwalk - - agent·Î °ÎÅÍ ¸ðµç SNMP objectµéÀ» °¡Á®¿Â´Ù.

Description

int snmpwalk(int hostname, string community, string object_id);

Returns an array of SNMP object values starting from the object_id as root and false on error.

snmpwalk() function is used to read all the values from an SNMP agent specified by the hostname. Community specifies the read community for that agent. A null object_id is taken as the root of the SNMP objects tree and all

objects under that tree are returned as an array. If object_id is specified, all the SNMP objects below that object_id are returned.

```
 $a = snmpwalk("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for($i=0; $i<count($a); $i++) {
        echo $a[$i];
}
```

snmprealwalk

snmprealwalk - - ³×Æ®¿÷ entity¿¡ ´ëÇÑ Á¤º¸ tree¸¦ ÁúÀÇÇÑ´Ù.

Description

array snmprealwalk(string hostname, string community, string object_id, int [timeout] , int [retries]);

Returns an associative array with object ids and their respective object value starting from the *object_id* as root and false on error.

snmprealwalk() function is used to read all object ids and their respective values from an SNMP agent specified by the hostname. Community specifies the read *community* for that agent. A null *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

The existence of **snmprealwalk()** and **snmpwalk()** has historical reasons. Both functions are provided for backward compatibility.

```
$a = snmprealwalk("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

XLII. String functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº ¹®ÀÚ¿-À» ´Ù·ç´Â ÇÔ¼öµéÀÌ´Ù. ´õ ÀÚ¼¼ÇÑ »çÇ×Àº regular expression ÀÌ³ª URL handling Àý¿¡ ¼³¸íµÈ ºÎºÐÀ» ÂüÁ¶ÇÏ¶ó.

AddSlashes

AddSlashes - - ÇÊ¿äÇÑ °÷À» backslash·¯ °¨½Ñ´Ù.

Description

string addslashes(string str);

Returns a string with backslashes before characters that need to be quoted in database queries etc. These characters are single quote ('), double quote ("), backslash (\) and NUL (the null byte).

See also **stripslashes()**, **htmlspecialchars()** and **quotemeta()**.

Chop

Chop - - µÚÂÊÀÇ ¿©¹é¸¦ »èÁ¦ÇÑ´Ù.

Description

string chop(string str);

Returns the argument string without trailing whitespace.
Example 1. chop() example

```
$trimmed = Chop($line);
```

See also **trim()**.

Chr

Chr - - ÁöÁ¤µÈ ¹®ÀÚ¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

string chr(int ascii);

Returns a one-character string containing the character specified by *ascii*.
Example 1. chr() example

```
$str .= chr(27); /* add an escape character at the end of $str */
/* Often this is more useful */
$str = sprintf("The string ends in escape: %c", 27);
```

This function complements **ord()**. See also **sprintf()** with a format string of %c.

chunk_split

chunk_split - - ¹®ÀÚ¿-À» º¯½ÁÇÑ Å©±â·Î ³ª´«´Ù.

Description

string chunk_split(string string, int [chunklen] , string [end]);

Can be used to split a string into smaller chunks which is useful for e.g. converting base64_encode output to match RFC 2045 behaviour. It inserts every *chunklen* (defaults to 76) chars the string *end* (defaults to " \ r \ n"). It returns the new string leaving the original string untouched.
Example 1. chr_replace() example

```
# format $data using RFC 2045 semantics
$new_string = chunk_split(base64_encode($data));
```

This function is significantly faster than **ereg_replace()**.

convert_cyr_string

convert_cyr_string - - ¹®ÀÚ¿-À» Æ¯Á¤ Cyrillic ¹®ÀÚ¿¡¼- ´Ù¸¥ °ÍÀ¸·Î ¹Ù²Û´Ù.

Description

string convert_cyr_string(string str, string from, string to);

This function converts the given string from one Cyrillic character set to another. The *from* and *to* arguments are single characters that represent the source and target Cyrillic character sets. The supported types are:

```
k - koi8-r
w - windows-1251
i - iso8859-5
a - x-cp866
d - x-cp866
m - x-mac-cyrillic
```

crypt

crypt - - ¹®ÀÚ¿-À» DES encryption¹æ¹ýÀ¸·Î º¯È¯(encrypt)ÇÑ´Ù.

Description

string crypt(string str, string [salt]);

crypt() will encrypt a string using the standard Unix DES encryption method. Arguments are a string to be encrypted and an optional two - character salt string to base the encryption on. See the Unix man page for your crypt function for more information.

If the salt argument is not provided, it will be randomly generated by PHP.

Some operating systems support more than one type of encryption. In fact, sometimes the standard DES encryption is replaced by an MD5 based encryption algorithm. The encryption type is triggered by the salt argument. At install time, PHP determines the capabilities of the crypt function and will accept salts for other encryption types. If no salt is provided, PHP will auto - generate a standard 2 - character DES salt by default unless the default encryption type on the system is MD5 in which case a random MD5 - compatible salt is generated.

The standard DES encryption **crypt()** contains the salt as the first two characters of the output.

There is no decrypt function, since **crypt()** uses a one - way algorithm.

echo

echo - - ÇÏ³ª ÀÌ»óÀÇ ¹®ÀÚ¿-À» Ãâ·ÂÇÑ´Ù.

Description

echo(string arg1, string [argn]...);

Outputs all parameters.

echo() is not actually a function (it is a language construct) so you are not required to use parantheses with it.

Example 1. echo example

```
echo "Hello World";
```

See also: **print() printf() flush()**

explode

explode - - ¹®ÀÚ¿-À» ¹®ÀÚ¿- ±âÁØÀ¸·Î ³ª´«´Ù.

Description

array explode(string separator, string string);

Returns an array of strings containing the elements separated by *separator*.
Example 1. explode() example

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
```

See also **split()** and **implode()**.

flush

flush - - output buffer¸¦ FlushÇÑ´Ù.

Description

void flush(void);

Flushes the output buffers of PHP and whatever backend PHP is using (CGI, a web server, etc.) This effectively tries to push all the output so far to the user's browser.

get_meta_tags

get_meta_tags - - ÆÄÀÏ¿¡¼- ¸ðµç meta tagÀÇ content ¼Ó¼º¸À» ÃßÃâÇÏ¿© ¹è¿-¿¡ ÀúÀåÇÑ´Ù.

Description

array get_meta_tags(string filename, int [use_include_path]);

Opens *filename* and parses it line by line for <meta> tags of the form
Example 1. Meta Tags Example

```
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!-- parsing stops here -->
```

(pay attention to line endings - PHP3 uses a native function to parse the input, so a Mac file won't work on Unix).

The value of the name property becomes the key, the value of the content property becomes the value of the returned array, so you can easily use standard array functions to traverse it or access single values. Special characters in the value of the name property are substituted with '_', the rest is converted to lower case.

Setting *use_include_path* to 1 will result in PHP3 trying to open the file along the standard include path.

htmlspecialchars

htmlspecialchars - - Æ¯¼ö¹®ÀÚ¸éÀ» HTML entity·Î º¯È¯ÇÑ´Ù.

Description

string htmlspecialchars(string string);

Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings. This function returns a string with these conversions made.

This function is useful in preventing user-supplied text from containing HTML markup, such as in a message board or guest book application.

At present, the translations that are done are:

'&' (ampersand) becomes '&'
'"' (double quote) becomes '"'
'<' (less than) becomes '<'
'>' (greater than) becomes '>'

Note that this functions does not translate anything beyond what is listed above. For full entity translation, see **htmlentities()**.

See also **htmlentities()** and **nl2br()**.

htmlentities

htmlentities - - ¸ðµç °¡´ÉÇÑ ¹®ÀÚµéÀ» HTML entity·Î ¹Ù²Û´Ù.

Description

string htmlentities(string string);

This function is identical to **htmlspecialchars()** in all ways, except that all characters which have HTML entity equivalents are translated into these entities.

At present, the ISO - 8859 - 1 character set is used.

See also **htmlspecialchars()** and **nl2br()**.

implode

implode - - ¹è¿-ÀÇ ¿ø¼ÒµéÀ» ¹®ÀÚ¿-À» »ç¿ëÇØ ¿¬°áÇÑ´Ù.

Description

string implode(array pieces, string glue);

Returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.
Example 1. implode() example

```
$colon_separated = implode($array, ":");
```

See also **explode()**, **join()**, and **split()**.

join

join - - ¹è¿-ÀÇ ¿ø¼ÒµéÀ» ¹®ÀÚ¿-À» »ç¿ëÇØ ¿¬°áÇÑ´Ù

Description

string join(array pieces, string glue);

join() is an alias to **implode()**, and is identical in every way.

ltrim

ltrim - - ¹®ÀÚ¿-ÀÇ ¾Õ ¿©¹éÀ» Á¦°ÅÇÑ´Ù.

Description

string ltrim(string str);

This function strips whitespace from the start of a string and returns the stripped string.

See also **chop()** and **trim()**.

md5

md5 - - ¹®ÀÚ¿-ÀÇ md5 hash °ªÀ» °è»êÇÑ´Ù.

Description

`string md5`(string str);

Calculates the MD5 hash of *str* using the RSA Data Security, Inc. MD5 Message - Digest Algorithm.

(¿ªÀÚÁÖ : MD5 algorithm¿¡ °üÇÑÀÚ¼¼ÇÑ ¼³¸íÀº RFC 1321À» º¸¼¼¿ä.)

nl2br

nl2br - - newlineÀ» HTML line break·Î º¯È¯ÇÑ´Ù.

Description

string nl2br(string string);

Returns *string* with '
' inserted before all newlines.

See also **htmlspecialchars()** and **htmlentities()**.

Ord

Ord - - ¹®ÀÚÀÇ ASCII °ªÀ» ±¸ÇÑ´Ù.

Description

int ord(string string);

Returns the ASCII value of the first character of *string*. This function complements **chr()**.
Example 1. ord() example

```
if (ord($str) == 10) {
    echo("The first character of \$str is a line feed.\n");
}
```

See also **chr()**.

parse_str

parse_str - - ¹®ÀÚ¿À» º¯¼ö·Î ÇØ¼®(parse)ÇÑ´Ù.

Description

`void parse_str`(string str);

Parses *str* as if it were the query string passed via an URL and sets variables in the current scope.
Example 1. Using parse_str()

```
$str = "first=value&second[]=this+works&second[]=another";
parse_str($str);
echo $first; /* prints "value" */
echo $second[0]; /* prints "this works" */
echo $second[1]; /* prints "another" */
```

print

print - - ¹®ÀÚ¿À» Ãâ·ÂÇÑ´Ù.

Description

print(string arg);

Outputs *arg*.

See also: **echo() printf() flush()**

printf

printf - - ¹®ÀÚ¿À» formatÇÏ¿© Ãâ·ÂÇÑ´Ù.

Description

int printf(string format, mixed [args]...);

Produces output according to *format*, which is described in the documentation for **sprintf()**.

See also: **print()**, **sprintf()**, and **flush()**.

quoted_printable_decode

quoted_printable_decode - - quoted - printable ¹®ÀÚ¿-À» 8 bit ¹®ÀÚ¿-·Î ¹Ù²Û´Ù.

Description

string quoted_printable_decode(string str);

This function returns an 8 - bit binary string corresponding to the decoded quoted printable string. This function is similar to **imap_qprint()**, except this one does not require the IMAP module to work.

QuoteMeta

QuoteMeta - - meta characterµéÀ» backslash°¡ ºÙÀº ¹®ÀÚ·Î ÀüÈ¯ÇÑ´Ù.

Description

int quotemeta(string str);

Returns a version of str with a backslash character (\) before every character that is among these:

. \\ + * ? [^] ($)

See also **addslashes()**, **htmlentities()**, **htmlspecialchars()**, **nl2br()**, and **stripslashes()**.

rawurldecode

rawurldecode - - URL - encoded ¹®ÀÚ¿-À» decodeÇÑ´Ù.

Description

string rawurldecode(string str);

Returns a string in which the sequences with percent (%) signs followed by two hex digits have been replaced with literal characters. For example, the string

foo%20bar%40baz

decodes into

foo bar@baz

See also **rawurlencode()**.

rawurlencode

rawurlencode - - RFC1738À» µû¶ó URL - encodeÇÑ´Ù.

Description

string rawurlencode(string str);

Returns a string in which all non - alphanumeric characters except

-_.

have been replaced with a percent (%) sign followed by two hex digits. This is the encoding described in RFC1738 for protecting literal characters from being interpreted as special URL delimiters, and for protecting URL's from being mangled by transmission media with character conversions (like some email systems). For example, if you want to include a password in an ftp url:

Example 1. rawurlencode() example 1

```
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'),
     '@ftp.my.com/x.txt">';
```

Or, if you pass information in a path info component of the url:
Example 2. rawurlencode() example 2

```
echo '<A HREF="http://x.com/department_list_script/',
     rawurlencode ('sales and marketing/Miami'), '">';
```

See also **rawurldecode()**.

setlocale

setlocale - - locale informationÀ» ¼³Á¤ÇÑ´Ù.

Description

`string setlocale`(string category, string locale);

category is a string specifying the category of the functions affected by the locale setting:

LC_ALL for all of the below
LC_COLLATE for string comparison - not currently implemented in PHP
LC_CTYPE for character classification and conversion, for example **strtoupper()**
LC_MONETARY for localeconv() - not currently implemented in PHP
LC_NUMERIC for decimal separator
LC_TIME for date and time formatting with **strftime()**

If *locale* is the empty string "", the locale names will be set from the values of environment variables with the same names as the above categories, or from "LANG".

If locale is zero or "0", the locale setting is not affected, only the current setting is returned.

Setlocale returns the new current locale, or false if the locale functionality is not implemented in the plattform, the specified locale does not exist or the category name is invalid. An invalid category name also causes a warning message.

similar_text

similar_text - - µÎ ¹®ÀÚ¿- °£ÀÇ °ñ½ÁÇÑ Á¤µµ¸¦ °è»êÇÑ´Ù.

Description

`int similar_text`(string first, string second, double [percent]);

This calculates the similarity between two strings as described in Oliver [1993]. Note that this implementation does not use a stack as in Oliver's pseudo code, but recursive calls which may or may not speed up the whole process. Note also that the complexity of this algorithm is O(N**3) where N is the length of the longest string.

By passing a reference as third argument, **similar_text()** will calculate the similarity in percent for you. It returns the number of matching chars in both strings.

soundex

soundex - - ¹®ÀÚ¿-ÀÇ soundex key¸¦ ±¸ÇÑ´Ù.(¿ªÀÚÁÖ: °ñ½ÁÇÑ ¹ßÀ½À» Ã£±â À§ÇØ »ç¿ë, ¿µ¾î¸¸ °¡´É)

Description

`string soundex`(string str);

Calculates the soundex key of *str*.

Soundex keys have the property that words pronounced similarly produce the same soundex key, and can thus be used to simplify searches in databases where you know the pronunciation but not the spelling. This soundex function returns a string 4 characters long, starting with a letter.

This particular soundex function is one described by Donald Knuth in "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison - Wesley (1973), pp. 391 - 392.
Example 1. Soundex Examples

```
soundex("Euler") == soundex("Ellery") == 'E460';
soundex("Gauss") == soundex("Ghosh") == 'G200';
soundex("Knuth") == soundex("Kant") == 'H416';
soundex("Lloyd") == soundex("Ladd") == 'L300';
soundex("Lukasiewicz") == soundex("Lissajous") == 'L222';
```

sprintf

sprintf - - formatµÈ ¹®ÀÚ¿-À» ¹ÝÈ¯ÇÑ´Ù.

Description

sprintf(string format, mixed [args]...);

Returns a string produced according to the formatting string *format*.

The format string is composed by zero or more directives: ordinary characters (excluding %) that are copied directly to the result, and *conversion specifications*, each of which results in fetching its own parameter. This applies to both **sprintf()** and **printf()**

Each conversion specification consists of these elements, in order:

An optional *padding specifier* that says what character will be used for padding the results to the right string size. This may be a space character or a 0 (zero character). The default is to pad with spaces. An alternate padding character can be specified by prefixing it with a single quote ('). See the examples below.

An optional *alignment specifier* that says if the result should be left - justified or right - justified. The default is right - justified; a - character here will make it left - justified.

An optional number, a *width specifier* that says how many characters (minimum) this conversion should result in.

An optional *precision specifier* that says how many decimal digits should be displayed for floating - point numbers. This option has no effect for other types than double. (Another function useful for formatting numbers is **number_format()**.)

A *type specifier* that says what type the argument data should be treated as. Possible types:
% - a literal percent character. No argument is required.
b - the argument is treated as an integer, and presented as a binary number.
c - the argument is treated as an integer, and presented as the character with that ASCII value.
d - the argument is treated as an integer, and presented as a decimal number.
f - the argument is treated as a double, and presented as a floating - point number.
o - the argument is treated as an integer, and presented as an octal number.
s - the argument is treated as and presented as a string.
x - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
X - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

See also: **printf()**, **number_format()**

Examples

Example 1. sprintf: zero - padded integers

```
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Example 2. sprintf: formatting currency

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf ("%01.2f", $money);
// echo $formatted will output "123.10"
```

strchr

strchr - - ÇØ´ç ¹®ÀÚ°¡ Ã³À½ ³ªÅ¸³ª´Â °÷À» Ã£´Â´Ù.

Description

string strchr(string haystack, string needle);

This function is an alias for **strstr()**, and is identical in every way.

strcmp

strcmp - - binary ¹æ½ÄÀÇ ¹®ÀÚ¿- °ñ±³

Description

int strcmp(string str1, string str2);

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also **ereg()**, **substr()**, and **strstr()**.

strcspn

strcspn - - mask¿¡ ¸ÂÁö ¾Ê´Â initial segmentÀÇ ±æÀÌ¸¦ ¸¸µçÙ.

Description

int strcspn(string str1, string str2);

Returns the length of the initial segment of *str1* which does *not* contain any of the characters in *str2*.

See also **strspn()**.

strip_tags

strip_tags - - ¹®ÀÚ¿-¿¡¼- HTML°ú PHP Å×±×µéÀ» Á¦°ÅÇÑ´Ù.

Description

string strip_tags(string str);

This function tries to strip all HTML and PHP tags from the given string. It errors on the side of caution in case of incomplete or bogus tags. It uses the same tag stripping state machine as the **fgetss()** function.

StripSlashes

StripSlashes - - addslashes·Î quoteµÈ ¹®ÀÚ¿-À» un - quoteÇÑ´Ù.

Description

string stripslashes(string str);

Returns a string with backslashes stripped off. (\' becomes ' and so on.) Double backslashes are made into a single backslash.

See also **addslashes()**.

strlen

strlen - - ¹®ÀÚ¿-ÀÇ ±æÀÌ¸¦ ¸¸µçÙ.

Description

int strlen(string str);

Returns the length of *string*.

strrpos

strrpos - - ¹®ÀÚ¿-¿¡¼- ÇØ´ç ¹®ÀÚ°¡ ³ªÅ¸³ª´Â Á¦ÀÏ ¸¶Áö¸· À§Ä¡¸¦ ¸¸µçÙ.

Description

string strrpos(string haystack, char needle);

Returns the numeric position of the last occurrence of *needle* in the *haystack* string. Note that the needle in this case can only be a single character. If a string is passed as the needle, then only the first character of that string will be used.

If *needle* is not found, returns false.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also **strpos()**, **strrchr()**, **substr()**, and **strstr()**.

strpos

strpos - - ÇØ´ç ¹®ÀÚ¿-ÀÌ ³ª¸¸Å¸³ª´Â Ã¹ À§Ä¡¸¦ ±¸Ñ´Ù.

Description

int strpos(string haystack, string needle, int [offset]);

Returns the numeric position of the first occurrence of *needle* in the *haystack* string. Unlike the **strrpos()**, this function can take a full string as the *needle* parameter and the entire string will be used.

If *needle* is not found, returns false.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

The optional *offset* parameter allows you to specify which character in *haystack* to start searching. The position returned is still relative to the the beginning of *haystack*.

See also **strrpos()**, **strrchr()**, **substr()**, and **strstr()**.

strrchr

strrchr - - ¹®ÀÚ¿-ÀÌ ¸¶Áö¸·À¸·Î ³ª¸¸Å¸³ª´Â À§Ä¡¸¦ ±¸Ñ´Ù.

Description

string strrchr(string haystack, string needle);

This function returns the portion of *haystack* which starts at the last occurrence of *needle* and goes until the end of *haystack*.

Returns false if *needle* is not found.

If *needle* contains more than one character, the first is used.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.
Example 1. strrchr() example

```
// get last directory in $PATH
$dir = substr( strrchr( $PATH, ":" ), 1 );
// get everything after last newline
$text = "Line 1\nLine 2\nLine 3";
$last = substr( strrchr( $text, 10 ), 1 );
```

See also **substr()** and **strstr()**.

strrev

strrev - - ¹®ÀÚ¿-À» ReverseÇÑ´Ù.

Description

string strrev(string string);

Returns *string*, reversed.

strspn

strspn - - mask¿¡ ¸Â´Â initial segmentÀÇ ±æÀÌ¸¦ ±¸Ñ´Ù.

Description

int strspn(string str1, string str2);

Returns the length of the initial segment of *str1* which consists entirely of characters in *str2*.

See also **strcspn()**.

strstr

strstr - - ¹®ÀÚ¿ Ã³À½ ³ª¸¸³ ´Â À§Ä¡ÀÇ ¹®ÀÚ¿» ¹ÝÈ¯ÇÑ´Ù.

Description

string strstr(string haystack, string needle);

Returns all of *haystack* from the first occurrence of *needle* to the end.

If *needle* is not found, returns false.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also **strrchr()**, **substr()**, and **ereg()**.

strtok

strtok - - ¹®ÀÚ¿À» tokenÈ- ÇÑ´Ù.

Description

string strtok(string arg1, string arg2);

strtok() is used to tokenize a string. That is, if you have a string like "This is an example string" you could tokenize this string into its individual words by using the space character as the token.
Example 1. strtok() example

```
$string = "This is an example string";
$tok = strtok($string," ");
while($tok) {
    echo "Word=$tok<br>";
    $tok = strtok(" ");
}
```

Note that only the first call to strtok uses the string argument. Every subsequent call to strtok only needs the token to use, as it keeps track of where it is in the current string. To start over, or to tokenize a new string you simply call strtok with the string argument again to initialize it. Note that you may put multiple tokens in the token parameter. The string will be tokenized when any one of the characters in the argument are found.

Also be careful that your tokens may be equal to "0". This evaluates to false in conditional expressions.

See also **split()** and **explode()**.

strtolower

strtolower - - ¼Ò¹®ÀÚ·Î ¸¸µç´Ù.

Description

string strtolower(string str);

Returns *string* with all alphabetic characters converted to lowercase.

Note that 'alphabetic' is determined by the current locale. This means that in i.e. the default "C" locale, characters such as umlaut-A (Ä) will not be converted.

See also **strtoupper()** and **ucfirst()**.

strtoupper

strtoupper - - ´ë¹®ÀÚ·Î ¸¸µç´Ù.

Description

string strtoupper(string string);

Returns *string* with all alphabetic characters converted to uppercase.

Note that 'alphabetic' is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

See also **strtolower()** and **ucfirst()**.

str_replace

str_replace - - Replace all occurrences of needle in haystack with str

Description

`string str_replace`(string needle, string str, string haystack);

This function replaces all occurences of *needle* in *haystack* with the given *str*. If you don't need fancy replacing rules, you should always use this function instead of **ereg_replace()**.
Example 1. str_replace() example

```
$bodytag = str_replace("%body%", "black", "<body text=%body%>");
```

This function is binary safe.

See also **ereg_replace()**.

strtr

strtr - - Æ¯Á¤ ¹®ÀÚ¿À» ´ëÄ¡(translate)ÇÑ´Ù.

Description

`string strtr`(string str, string from, string to);

This function operates on *str*, translating all occurrences of each character in *from* to the corresponding character in *to* and returning the result.

If *from* and *to* are different lengths, the extra characters in the longer of the two are ignored.
Example 1. strtr() example

```
$addr = strtr($addr, "äåö", "aao");
```

See also **ereg_replace()**.

substr

substr - - ¹®ÀÚ¿Ç ÀÏºÎºÐÀ» ¹ÝÈ¯ÇÑ´Ù.

Description

`string substr`(string string, int start, int [length]);

Substr returns the portion of *string* specified by the *start* and *length* parameters.

If *start* is positive, the returned string will start at the *start*'th character of *string*. Examples:

```
$rest = substr("abcdef", 1); // returns "bcdef"
$rest = substr("abcdef", 1, 3); // returns "bcd"
```

If *start* is negative, the returned string will start at the *start*'th character from the end of *string*. Examples:

```
$rest = substr("abcdef", -1); // returns "f"
$rest = substr("abcdef", -2); // returns "ef"
```

```
$rest = substr("abcdef", -3, 1); // returns "d"
```

If *length* is given and is positive, the string returned will end *length* characters from *start*. If this would result in a string with negative length (because the start is past the end of the string), then the returned string will contain the single character at *start*.

If *length* is given and is negative, the string returned will end *length* characters from the end of *string*. If this would result in a string with negative length, then the returned string will contain the single character at *start*. Examples:

```
$rest = substr("abcdef", -1, -1); // returns "bcde"
```

See also **strrchr()** and **ereg()**.

trim

trim - - ¹®ÀÚ¿-ÀÇ ¾Õ,µÚ ¿©¹é ¸ðµÎ¸¦ Á¦°ÅÇÑ´Ù.

Description

string trim(string str);

This function strips whitespace from the start and the end of a string and returns the stripped string.

See also **chop()** and **ltrim()**.

ucfirst

ucfirst - - ¹®ÀÚ¿-ÀÇ Ã³À½ ¹®ÀÚ¸¦ ´ë¹®ÀÚ·Î ¸¸µç´Ù.

Description

string ucfirst(string str);

Capitalizes the first character of *str* if that character is alphabetic.

Note that 'alphabetic' is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

See also **strtoupper()** and **strtolower()**.

ucwords

ucwords - - Uppercase the first character of each word in a string

Description

```
string ucwords(string str);
```

Capitalizes the first character of each word in *str* if that character is alphabetic.

See also **strtoupper()**, **strtolower()** and **ucfirst()**.

XLIII. URL functions

Table of Contents

parse_url

parse_url - - PHP°¡ form data¸¦ ÇØ¼®ÇÏµíÀÌ query stringÀ» ÇØ¼®(parse)ÇÑ´Ù.

Description

array parse_url(string url);

This function returns an associative array returning any of the various components of the URL that are present. This includes the "scheme", "host", "port", "user", "pass", "path", "query", and "fragment".

urldecode

urldecode - - URL - encodeµÈ ¹®ÀÚ¿-À» decodeÇÑ´Ù.

Description

string urldecode(string str);

Decodes any %## encoding in the given string. The decoded string is returned.
Example 1. urldecode() example

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
  $b = split ('=', $a [$i]);
  echo 'Value for parameter ', htmlspecialchars (urldecode ($b [0])),
       ' is ', htmlspecialchars (urldecode ($b [1])), "<BR>";
  $i++;
}
```

See also **urlencode()**

urlencode

urlencode - - ¹®ÀÚ¿-À» URL - encodeÇÑ´Ù.

Description

string urlencode(string str);

Returns a string in which all non - alphanumeric characters except -_. have been replaced with a percent (%) sign followed by two hex digits and spaces encoded as plus (+) signs. It is encoded the same way that the posted data from a WWW form is encoded, that is the same way as in application/x-www-form-urlencoded media type. This differs from the RFC1738 encoding (see **rawurlencode()**) in that for historical reasons, spaces are encoded as plus (+) signs. This function is convenient when encoding a string to be used in a query part of an URL, as a convinient way to pass variables to the next page:
Example 1. urlencode() example

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '">';
```

See also **urldecode()**

base64_encode

base64_encode - - base64 ¹æ½ÄÀ¸·Î encodeÇÑ´Ù.

Description

string base64_encode(string data);

base64_encode() returns *data* encoded with base64. This encoding is designed to make binary data survive transport through transport layers that are not 8 - bit clean, such as mail bodies.

Base64 - encoded data takes about 33% more space than the original data.

See also: **base64_decode()**, RFC - 2045 section 6.8.

base64_decode

base64_decode - - base64·Î encodeµÈ ¹®ÀÚ¿-À» decode ÇÑ´Ù.

Description

string base64_decode(string encoded_data);

base64_decode() decodes *encoded_data* and returns the original data. The returned data may be binary.

See also: **base64_encode()**, RFC - 2045 section 6.8.

XLIV. Variable functions

Table of Contents

gettype

gettype - - º¯¼öÀÇ typeÀ» ±¸ÇÑ´Ù.

Description

string gettype(mixed var);

Returns the type of the PHP variable *var*.

Possibles values for the returned string are:

"integer"
"double"
"string"
"array"
"class"
"object"
"unknown type"

See also **settype()**.

intval

intval - - º¯¼öÀÇ Á¤¼ö°ªÀ» ±¸ÇÑ´Ù.

Description

int intval(mixed var, int [base]);

Returns the integer value of *var*, using the specified base for the conversion (the default is base 10).

var may be any scalar type. You cannot use **intval()** on arrays or objects.

See also **doubleval()**, **strval()**, **settype()** and Type juggling.

doubleval

doubleval - - º¯¼öÀÇ ½Ç¼ö(double)°ªÀ» ±¸ÇÑ´Ù.

Description

double doubleval(mixed var);

Returns the double (floating point) value of *var*.

var may be any scalar type. You cannot use **doubleval()** on arrays or objects.

See also **intval()**, **strval()**, **settype()** and Type juggling.

empty

empty - - º¯¼ö°¡ °ªÀÌ ºñ¾î ÀÖ´Â°¡ °Ë»çÇÑ´Ù.

Description

int empty(mixed var);

Returns false if *var* exists and has a non-empty or non-zero value; true otherwise.

See also **isset()** and **unset()**.

is_array

is_array - - º¯¼ö°¡ ¹è¿ÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_array(mixed var);

Returns true if *var* is an array, false otherwise.

See also **is_double()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, and **is_object()**.

is_double

is_double - - º¯¼ö°¡ ½Ç¼öÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_double(mixed var);

Returns true if *var* is a double, false otherwise.

See also **is_array()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, and **is_object()**.

is_float

is_float - - º¯¼ö°¡ ½Ç¼öÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_float(mixed var);

This function is an alias for **is_double()**.

See also **is_double()**, **is_real()**, **is_int()**, **is_integer()**, **is_string()**, **is_object()**, **is_array()**, and **is_long()**.

is_int

is_int - - º¯¼ö°¡ Á¤¼öÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_int(mixed var);

This function is an alias for **is_long()**.

See also **is_double()**, **is_float()**, **is_integer()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, and **is_long()**.

is_integer

is_integer - - º¯¼ö°¡ Á¤¼öÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_integer(mixed var);

This function is an alias for **is_long()**.

See also **is_double()**, **is_float()**, **is_int()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, and **is_long()**.

is_long

is_long - - º¯¼ö°¡ Á¤¼öÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_long(mixed var);

Returns true if *var* is an integer (long), false otherwise.

See also **is_double()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, and **is_integer()**.

is_object

is_object - - º¯¼ö°¡ ´Ã¼(object)ÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_object(mixed var);

Returns true if *var* is an object, false otherwise.

See also **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_string()**, and **is_array()**.

is_real

is_real - - º¯¼ö°¡ ½Ç¼öÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_real(mixed var);

This function is an alias for **is_double()**.

See also **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_object()**, **is_string()**, and **is_array()**.

is_string

is_string - - º¯¼ö°¡ ¹®ÀÚ¿ÀÎ°¡ °Ë»çÇÑ´Ù.

Description

int is_string(mixed var);

Returns true if *var* is a string, false otherwise.

See also **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_object()**, and **is_array()**.

isset

isset - - º¯¼ö¿¡ °ªÀÌ ÀÖ´Â°¡ °Ë»çÇÑ´Ù.

Description

int isset(mixed var);

Returns true if *var* exists; false otherwise.

If a variable has been unset with **unset()**, it will no longer be **isset()**.

```
$a = "test";
echo isset($a); // true
unset($a);
echo isset($a); // false
```

See also **empty()** and **unset()**.

settype

settype - - º¯¼öÀÇ typeÀ» ¼³Á¤ÇÑ´Ù.

Description

int settype(string var, string type);

Set the type of variable *var* to *type*.

Possibles values of *type* are:

"integer"
"double"
"string"
"array"
"object"

Returns true if successful; otherwise returns false.

See also **gettype()**.

strval

strval - - º¯¼öÀÇ °ªÀ» ¹®ÀÚ¿·Î ¸¸µç´Ù.

Description

string strval(mixed var);

Returns the string value of *var*.

var may be any scalar type. You cannot use **strval()** on arrays or objects.

See also **doubleval()**, **intval()**, **settype()** and Type juggling.

unset

unset - - º¯¼ö¸¦ UnsetÇÑ´Ù.

Description

int unset(mixed var);

unset() destroys the specified variable and returns true.
Example 1. unset() example

```
unset( $foo );
unset( $bar['quux'] );
```

See also **isset()** and **empty()**.

XLV. Vmailmgr Functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº QMAIL (www.qmail.org) °ú vmailmgr ÆÐÅ°Áö¸¦ ÇÊ¿ä·Î ÇÑ´Ù. ÀÌ ÆÐÅ°Áö´Â Bruce Guenter¿¡ ÀÇÇØ Á¦°øµÇ°í ÀÖ°í http://www.qcc.sk.ca/~bguenter/distrib/vmailmgr/ ¿¡¼ Ã£À» ¼ö ÀÖ´Ù.

¾Æ·¡ÀÇ ÇÔ¼öµé¿¡¼´Â µÎ °³ÀÇ °¯¼ö°¡ Á¤ÀÇµÇ¾î ÀÖ´Ù : ¿©·¯°ÐÀÇ °¡»ó domain (vdomain.com)ÀÇ µµ¸ÞÀÎ ÀÌ¸§À» °ªÀ¸·Î °¡Áö´Â vdomainÀÌ¶ó´Â string °¯¼ö¿Í, °¡»ó userµéÀ» °¡Áö°í ÀÖ´Â 'ÁøÂ¥(real)' userÀÇ password¸¦ °ªÀ¸·Î °¡Áö´Â basepwd¶ó´Â string °¯¼öÀÌ´Ù.

°¡»ó userµéÀÇ password´Â 8ÀÚ±îÁö¸¸ ÀÎ½ÄµÇ°í ±× ÀÌÈÄ´Â ¹«½ÃµÈ´Ù.

¸ðµç ÇÔ¼öµéÀÇ return »óÅÂ´Â response.h ¿¡ ´ÙÀ½°ú °°ÀÌ Á¤ÀÇµÇ¾î ÀÖ´Ù.
O ok
1 bad
2 error
3 error connecting

¾Ë·ÁÁø ¹®Á¦µé: **vm_deluser()**´Â »ç¿ëÀÚÀÇ directory¸¦ Áö¿ìÁö ¾Ê´Â´Ù. **vm_addalias()**´Â ÇöÀç Á¦´ë·Î µ¿ÀÛÇÏÁö ¾Ê´Â´Ù.

```
<?php
dl("php3_vmailmgr.so"); //load the shared library
$vdomain="vdomain.com";
$basepwd="password";
?>
```

vm_adduser

vm_adduser - - ÁÖ¾îÁø password·Î »õ virtual user¸¦ ¸¸µç´Ù.

Description

int vm_adduser(string vdomain, string basepwd, string newusername, string newuserpassword);

Add a new virtual user with a password. *newusername* is the email login name and *newuserpassword* the password for this user.

vm_addalias

vm_addalias - - virtual user¿¡ alias¸¦ Ãß°¡ÇÑ´Ù.

Description

int vm_addalias(string vdomain, string basepwd, string username, string alias);

Add an alias to a virtual user. *username* is the email login name and *alias* is an alias for this vuser.

vm_passwd

vm_passwd - - virtual userÀÇ password¸¦ °¯°æÇÑ´Ù.

Description

int vm_passwd(string vdomain, string username, string password, string newpassword);

Changes a virtual users password. *username* is the email login name, *password* the old password for the vuser, and *newpassword* the new password.

vm_delalias

vm_delalias - - alias¸¦ Áö¿î´Ù.

Description

int vm_delalias(string vdomain, string basepwd, string alias);

Removes an alias.

vm_deluser

vm_deluser - - virtual user¸¦ »èÁ¦ÇÑ´Ù.

Description

int vm_deluser(string vdomain, string username);

Removes a virtual user.

XLVI. WDDX functions

Table of Contents

ÀÌ ÇÔ¼öµéÀº WDDX¿Í ÇÔ²² µ¿ÀÛÇÏµµ·Ï µÇ¾îÀÖ´Ù.

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.

Example 1. Serializing a single value

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

This example will produce:

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

Example 2. Using incremental packets

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");
/* Suppose $cities came from database */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");
$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

This example will produce:

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

wddx_serialize_value

wddx_serialize_value - - WDDX ÆÐÄÏ¿¡ ´ÜÀÏ °ªÀ» serialize ÇÑ´Ù.

Description

string wddx_serialize_value(mixed var, string [comment]);

wddx_serialize_value() is used to create a WDDX packet from a single given value. It takes the value contained in *var*, and an optional *comment* string that appears in the packet header, and returns the WDDX packet.

wddx_serialize_vars

wddx_serialize_vars - - WDDX ÆÐÄÏ¿¡ ¿©·¯ °¹ö¸¦ serialize ÇÑ´Ù.

Description

string wddx_serialize_vars(string var_name | array var_names [, ...]);

wddx_serialize_vars() is used to create a WDDX packet with a structure that contains the serialized representation of the passed variables.

wddx_serialize_vars() takes a variable number of arguments, each of which can be either a string naming a variable or an array containing strings naming the variables or another array, etc.
Example 1. wddx_serialize_vars example

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";
$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

The above example will produce:

```
<wddxPacket version='0.9'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

wddx_packet_start

wddx_packet_start - - ³»°Î¿¡ ÀÖ´Â ±¸Á¶´ë·Î »õ·Î¿î WDDX ÆÐÄÏÀ» ¸¸µç´Ù.

Description

int wddx_packet_start(string [comment]);

Use **wddx_packet_start()** to start a new WDDX packet for incremental addition of variables. It takes an optional *comment* string and returns a packet ID for use in later functions. It automatically creates a structure definition inside the packet to contain the variables.

wddx_packet_end

wddx_packet_end - - ÁöÁ¤µÈ IDÀÇ WDDX ÆÐÄÏÀ» Á¾·áÇÑ´Ù.

Description

int wddx_packet_end(int packet_id);

wddx_packet_end() ends the WDDX packet specified by the *packet_id* and returns the string with the packet.

wddx_add_vars

wddx_add_vars - - ÇØ´çÇÏ´Â ÆÐÄÏ¿¡ Ãß°¡·Î °¢¼öµéÀ» serializeÇÑ´Ù.

Description

wddx_add_vars(int packet_id, ...);

wddx_add_vars() is used to serialize passed variables and add the result to the packet specified by the *packet_id*. The variables to be serialized are specified in exactly the same way as **wddx_serialize_vars()**.

wddx_deserialize

wddx_deserialize - - WDDX ÆÐÄÏÀ» deserialize ÇÑ´Ù.

Description

mixed wddx_deserialize(string packet);

wddx_deserialized() takes a *packet* string and deserializes it. It returns the result which can be string, number, or array. Note that structures are deserialized into associative arrays.

XLVII. Gz - file Functions

Table of Contents

ÀÌ ¸ðµâÀº gzip(.gz)À¸·Î ¾ÐÃàµÈ ÆÄÀÏÀ» Åõ¸íÇÏ°Ô(¾ÐÃàµÇÁö ¾ÊÀº °ÍÃ³·³) ÀÐ°í ¾²±â À§ÇØ Jean-loup Gailly¿Í Mark Adler¿¡ ÀÇÇÑ zlib >= 1.0.9 (http://www.cdrom.com/pub/infozip/zlib/) ÇÔ¼öµéÀ» »ç¿ëÇÑ´Ù.

gzclose

gzclose -- gz-file pointer¸¦ ´Ý´Â´Ù.

Description

int gzclose(int zp);

The gz-file pointed to by zp is closed.

Returns true on success and false on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

gzeof

gzeof -- gz-file pointer°¡ end-of-file¿¡ ÀÖ´Â°¡ °Ë»çÇÑ´Ù.

Description

int gzeof(int zp);

Returns true if the gz-file pointer is at EOF or an error occurs; otherwise returns false.

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

gzfile

gzfile -- gz-fileÀÇ ³»¿ëÀ» ¹è¿·Î ÀÐ´Â´Ù.

Description

array gzfile(string filename);

Identical to **readgzfile()**, except that gzfile() returns the file in an array.

See also **readgzfile()**, and **gzopen()**.

gzgetc

gzgetc -- gz-file pointer¿¡¼ ÇÑ ¹®ÀÚ¸¦ ÀÐ´Â´Ù.

Description

string gzgetc(int zp);

Returns a string containing a single (uncompressed) character read from the file pointed to by zp. Returns FALSE

on EOF (as does **gzeof()**).

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, and **gzgets()**.

gzgets

gzgets - - gz-file pointer¿¡¼- ÇÑ ÁÙÀ» ÀÐ´Â´Ù.

Description

string gzgets(int zp, int length);

Returns a (uncompressed) string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, and **gzgetc()**.

gzgetss

gzgetss - - gz-file pointer¿¡¼- ÇÑ ÁÙÀ» ÀÐ¾î HTML tagµéÀ» stripÇÑ´Ù.

Description

string gzgetss(int zp, int length);

Identical to **gzgets()**, except that gzgetss attempts to strip any HTML and PHP tags from the text it reads.

See also **gzgets()**, and **gzopen()**.

gzopen

gzopen - - gz-fileÀ» ¿¬´Ù.

Description

int gzopen(string filename, string mode);

Opens a gzip (.gz) file for reading or writing. The mode parameter is as in **fopen()** ("rb" or "wb") but can also include a compression level ("wb9") or a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman only compression as in "wb1h". (See the description of deflateInit2 in zlib.h for more information about the strategy parameter.)

Gzopen can be used to read a file which is not in gzip format; in this case gzread will directly read from the file without decompression.

Gzopen returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns false.
Example 1. gzopen() example

```
$fp = gzopen("/tmp/file.gz", "r");
```

See also **gzclose()**.

gzpassthru

gzpassthru - - gz-file pointer¿¡¼-ºÎÅÍ µÚ¿¡ ³²¾ÆÀÖ´Â ¸ðµç µ¥ÀÌÅ¸¸¦ Ãâ·Â(output)ÇÑ´Ù.

Description

int gzpassthru(int zp);

Reads to EOF on the given gz-file pointer and writes the (uncompressed) results to standard output.

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

The gz-file is closed when **gzpassthru()** is done reading it (leaving *zp* useless).

gzputs

gzputs -- gz-file pointer¿¡ ¾´´Ù.

Description

int gzputs(int zp, string str, int [length]);

gzputs() is an alias to **gzwrite()**, and is identical in every way.

gzread

gzread -- gz-file¿¡¼- ÀÌÁø µ¥ÀÌÅ¸·Î ÀÐ´Â´Ù.

Description

string gzread(int zp, int length);

gzread() reads up to *length* bytes from the gz-file pointer referenced by *zp*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen( $filename, "r" );
$contents = gzread( $zd, 10000 );
gzclose( $zd );
```

See also **gzwrite()**, **gzopen()**, **gzgets()**, **gzgetss()**, **gzfile()**, and **gzpassthru()**.

gzrewind

gzrewind -- gz-file pointerÀÇ À§Ä¡¸¦ ¸ÇÃ³À½Î °¡Á®°£´Ù.

Description

int gzrewind(int zp);

Sets the file position indicator for zp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzseek()** and **gztell()**.

gzseek

gzseek -- gz-file pointer¸¦ ¿Å±ä´Ù.

Description

int gzseek(int zp, int offset);

Sets the file position indicator for the file referenced by zp to offset bytes into the file stream. Equivalent to calling (in C) `gzseek( zp, offset, SEEK_SET )`.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek then compresses a sequence of zeroes up to the new starting position.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

See also **gztell()** and **gzrewind()**.

gztell

gztell - - gz - file pointerÀÇ ÀÐ°í, ¾µ À§Ä¡¸¦ ¹ÝÈ¯ÇÑ´Ù.

Description

int gztell(int zp);

Returns the position of the file pointer referenced by zp; i.e., its offset into the file stream.

If an error occurs, returns false.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, **gzseek()** and **gzrewind()**.

readgzfile

readgzfile - - gz - fileÀ» ÃâÂ·Â(output)ÇÑ´Ù.

Description

int readgzfile(string filename);

Reads a file, decompresses it and writes it to standard output.

Returns the number of (uncompressed) bytes read from the file. If an error occurs, false is returned and unless the function was called as @readgzfile, an error message is printed.

The file *filename* will be opened from the filesystem and its contents written to standard output.

See also **gzpassthru()**, **gzfile()**, and **gzopen()**.

gzwrite

gzwrite - - gz - file¿¡ ÀÌÁø µ¥ÀÌÅ¸¸¦ ¾´´Ù.

Description

int gzwrite(int zp, string string, int [length]);

gzwrite() writes the contents of *string* to the gz - file stream pointed to by *zp*. If the *length* argument is given, writing will stop after *length* (uncompressed) bytes have been written or the end of *string* is reached, whichever comes first.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

See also **gzread()**, **gzopen()**, and **gzputs()**.

XLVIII. XML Parser Functions

Table of Contents

¼Ò°³ (Introduction)

XML¿¡ ´ëÇÏ¿© (About XML)

XML(eXtensible Markup Language)ÀÌ¶õ À¥¿¡¼ ±³È¯µÇ´Â ¹®¼¸¦ À§ÇÑ µ¥ÀÌÅÍ Æ÷¸ËÀç ÇÏ³ªÀÌ´Ù. ÀÌ°ÍÀº W3C(World Wide Web consortium)¿¡ ÀÇÇØ ±ÔÁ¤µÈ Ç¥ÁØÀÌ´Ù. XML¿¡ ´ëÇÑ Á¤º¸¿Í ±×¿¡ °ü·ÃµÈ ±â¼úµéÀº http://www.w3.org/XML/¿¡¼ Ã£¾Æ¾ ¼ö ÀÖ» °ÍÀÌ´Ù.

¼³Ä¡ (Installation)

ÀÌ È®ÀåÀº http://www.jclark.com/xml/¿¡¼ Ã£À» ¼ö ÀÖ´Â **expat**¶ó´Â °ÍÀ» »ç¿ëÇÏ°í ÀÖ´Ù. expact¿Í ÇÔ²² µû¶ó¿À´Â MakefileÀº ±×´ë·Î ¶óÀÌºê·¯¸®¸¦ ¸¸µé¾î ³»Áö ¸ØÇÑ´Ù. ´ÙÀ½°ú °°À° make ruleÀ» »ç¿ëÇØ¾ß °¡´ÉØ Áø´Ù. :

```
libexpat.a: $(OBJS)
        ar -rc $@ $(OBJS)
        ranlib $@
```

expactÀÇ ¼Ò½º RPM ÆÐÅ°Áö´Â http://www.guardian.no/~ssb/phpxml.html¿¡¼ Ã£À» ¼ö ÀÖ´Ù.

À¯´Ð½¿¡¼´Â **configure**¸¦ *--with-xml* ¿É¼ÇÀ» ÁÖ¾î ½ÇÇàÇÑ´Ù. **expat** ¶óÀÌºê·¯¸®´Â ¿©·¯°ÐÀÇ ÄÄÆÄÀÏ·¯°¡ ±â°»ÀûÀ¸·Î Ã£À» ¼ö ÀÖ´Â °÷¿¡ ¾³Ä¡µÇ¾î ÀÖ¾î¾ß ÇÑ´Ù. ¸¸à ¼³Ä¡ÇÑ expact°¡ ÄÄÆÄÀÏ·¯°¡ Ã£À» ¼ö ¾ø´Â °÷¿¡ ÀÖ´Ù¸é configure¸¦ ½ÇÇàÇÏ±â Àü¿¡ È¯°æº¯¼ö CPPFLAGS¿Í LDFLAGS¸¦ ¿©·¯°ÐÀÇ È¯°æ¿¡¼ Ç Ø´ç °ªÀ¸·Î ¼³Á¤ÇÒ ÇÊ¿ä°¡ ÀÖ´Ù.

ÀÌÁ¦ PHP¸¦ ÄÄÆÄÀÏÇÏ¸é µÈ´Ù.

ÀÌ È®Àå¿¡ ´ëÇÏ¿© (About This Extension)

ÀÌ È®ÀåÀº PHP¿¡¼ James ClarkÀÇ **expat**¸¦ Áö¿øÇÏµµ·Ï Ç Ø ÁØ´Ù. ÀÌ ToolkitÀº ¿©·¯°ÐÀÌ XMLÀ» Ç Ø¼®(parse)ÇÏµµ·Ï ÇÏÁö¸¸, À¯È¿ÇÑ°¡(validate)´Â È®ÀÎÇÏÁö ¾Ê´Â´Ù. ÀÌ È®ÀåÀº PHP°¡ Á¦°øÇÏ´Â ¼¼°¡Áö ¹®ÀÚ ÇüÅÂ(character encodings)¸¦ Áö¿øÇÑ´Ù. US-ASCII¿Í ISO-8859-1, UTF-8ÀÌ °¡´ÉÏ´Ù. UTF-16Àº Áö¿øÇÏÁö ¾Ê´Â´Ù.

ÀÌ È®ÀåÀ¸·Î ¿©·¯°ÐÀº XML ÆÄ¼¸¦ ¸¸µé°í(create XML parsers), ¼·Î´Ù¸¥ XML event¿¡ ´ëÇØ *handlers*¸¦ Á¤ÀÇÑ´Ù. °¢°¢ ÀÇ XML ÆÄ¼´Â ¿©·¯°ÐÀÌ Á¶Á¤°¡´ÉÑ ¸î °³ÀÇ parameters¸¦ °¡Áö°í ÀÖ´Ù.

Á¦øµÇ´Â XML event handlers´Â ´ÙÀ½°ú °°´Ù. :

Table 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 °ãÄ¡±â (Case Folding)

element handler ÇÔ¼öµéÀº ±×µéÀÇ ¿ä¼Ò¸¦ *case-folded*ÇÏ°Ô ¾ò´Â´Ù. Case-foldingÀº XML Ç¥ÁØ¿¡ ´ÙÀ½°ú °°ÀÌ Á¤ÀÇ µÇ¾î ÀÖ´Ù. : "a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents" (ÀÏ·ÃÀÇ ¹®ÀÚµé¿¡¼ ´ë¹®ÀÚ°¡ ¾Æ´Ñ ±ÛÀÚµéÀ» ´ë¹®ÀÚ¿Í µ¿ÀÏÇÏ°Ô ´Ù·ç´Â ¹æ¹ý).

´Ù¸¥ ¸»·Î ÇÏ¸é case-foldingÀÌ¶õ ´Ü¼øÈ÷ uppercasingÀ» ÀÇ¹ÌÇÑ´Ù.

±â°»ÀûÀ¸·Î, handler ÇÔ¼ö¿¡ Àü´ÞµÇ´Â ¸ðµç ¿ä¼ÒµéÀÇ ÀÌ¸§Àº case-foldedÇÏ´Ù. case-folded ¼³Á¤Àº **xml_parser_get_option()**°ú **xml_parser_set_option()** ÇÔ¼ö·Î XML ÆÄ¼¿¡ ´ëÇØ ÁúÀÇÇÏ°Å³ª ¿øÇÏ´Â ´ë·Î ¼³Á¤ÇÒ ¼ö ÀÖ´Ù.

Error Codes

´ÙÀ½ÀÇ »ó¼öµéÀÌ XML ¿¡·¯ ÄÚµå·Î Á¤ÀÇµÇ¾î ÀÖ´Ù. (**xml_parse()**ÀÇ ¹ÝÈ¯°ªÀÌ´Ù.)
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

PHPÀÇ XML È®ÀåÀº ¼·Î ´Ù¸¥ *character encoding*µéÀ» À§ÇØ Unicode character setÀ» Áö¿øÇÑ´Ù. character encoding¿¡´Â *source encoding*°ú *target encoding*ÀÇ µÎ°¡Áö À¯ÇüÀÌ ÀÖ´Ù. Âü°í·Î PHP´Â ³»°ÎÀûÀ¸·Î ¸ðµç ¹®¼¸¦ UTF-8À¸·Î encodeÇÏ°í ÀÖ´Ù.

Source encodingÀº XML ¹®¼°¡ ÇØ¼®(parse)µÉ ¶§ ÀÏ¾î³´Ù. XML ÆÄ¼¸¦ ¸¸µé ¶§(Upon creating an XML parser), source encodingÀº ÁöÁ¤µÉ ¼ö ÀÖ´Ù. (ÀÌ encodingÀº ÇØ´ç XML ÆÄ¼¿¡ ¾ø¾îÁú ¶§ ±îÁö °¨æµÉ ¼ö ¾ø´Ù.) Áö¿øµÇ´Â source encodingÀº ISO-8859-1¿Í US-ASCII, UTF-8ÀÇ ¼¼°¡Áö ÀÌ´Ù. ¾ÕÀÇ µÎ°¡Áö´Â single-byte encodingÀÌ´Ù. ÀÌ°ÍÀº °¢°¢ÀÇ ¹®ÀÚ°¡ 1°³ÀÇ byte·Î Ç¥ÇöµÈ´Ù´Â ÀÇ¹ÌÀÌ´Ù. UTF-8Àº 1 ¹ÙÀÌÆ®¿¡¼ 4 ¹ÙÀÌÆ®(21°ñÆ®)±îÁö °¨æ°¡´ÉÇÑ ¼öÀÇ °ñÆ®¼ö·Î ±¸°µÈ ¹®ÀÚ·Î encodeÇÑ´Ù. PHP¿¡¼ ±âº» source encodingÀº ISO-8859-1ÀÌ´Ù.

Target encodingÀº PHP°¡ XML handler ÇÔ¼ö·Î µ¥ÀÌÅÍ¸¦ ³Ñ°ÜÁÙ ¶§ ÀÏ¾î³´Ù. XML ÆÄ¼°¡ »ý¼°µÉ ¶§ target encodingÀº source encoding°ú °°Àº °ªÀ¸·Î ¼³Á¤µÇÁö¸¸, ±× °ªÀº ³ªÁß¿¡ ¾ðÁ¦¶óµµ ¹Ù²Ü ¼ö ÀÖ´Ù. target encodingÀº ¹®ÀÚ µ¥ÀÌÅÍ»Ó ¾Æ´Ï¶ó, tag ÀÌ¸§ÀÌ³ª processing instruction targets¿¡µµ Âû¿ëµÈ´Ù.

¸¸¾à XML ÆÄ¼°¡ ±×°ÍÀÇ source encodingÀÌ Ç¥ÇöÇÒ ¼ö ¾ø´Â ¹üÀ§¿¡ ÀÖ´Â ¹®ÀÚ¸¦ ¸¸³´Ù¸é, ÆÄ¼´Â ¿¡·¯¸¦ ¹ÝÈ¯ÇÑ´Ù.

PHP°¡ ÇØ¼®µÈ(parsed) ¹®¼Áß¿¡¼ ¼±ÅÃµÈ target encoding¾È¿¡¼ Ç¥ÇöÀÌ °Ò°¡´ÉÇÑ ¹®ÀÚ¸¦ ¸¸³´Ù¸é, ÀÌ ¹®Á¦°¡ µÈ ¹®ÀÚ¸¦ "demoted"ÇÑ´Ù. ÇöÀç ±×·± ¹®ÀÚµéÀº ¹Ù¸Ç¥(?)·Î ´ëÄ¡µÈ´Ù.

Some Examples

¿©±â¿¡ XML ¹®¼¸¦ ÇØ¼®ÇÏ´Â PHP ½ºÅ©¸³Æ® ¿¹Á¦°¡ ÀÖ´Ù.

XML Element Structure Example

¹®¼¿¡¼ ½ÃÀÛÀ¸ ¿ä¼ÒµéÀÇ ±¸Á¶¸¦ indentationÇÏ¿© Ç¥½ÃÇÏ´Â ¿¹Á¦. :
Example 1. Show XML Element Structure

```
$file = "data.xml";
$depth = array();
function startElement($parser, $name, $attrs)
{
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
```

```
}
function endElement($parser, $name, $attrs)
{
    global $depth;
    $depth[$parser]--;
}
$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML Tag Mapping Example

Example 2. Map XML to HTML

ÀÌ ¿¹Á¦´Â XML¹®¼ÀÇ ÅÃÀ» Á÷Á¢ÀûÀ¸·Î HTML ÅÃÀ¸·Î mapÇÑ´Ù. "map array"¿¡¼ Ã£À» ¼ö ¾ø´Â ¿ä¼Ò´Â ¹«½ÃµÈ´Ù. ¹°·Ð, ÀÌ ¿¹Á¦´Â Æ¯Á¤ÇÑ ¹®¼ Å¸ÀÔ¿¡¼¸¸ µ¿ÀÛÇÑ´Ù.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);
function startElement($parser, $name, $attrs)
{
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}
function endElement($parser, $name, $attrs)
{
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}
function characterData($parser, $data)
{
    print $data;
}
$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML External Entity Example

ÀÌ ¿¹Á¦´Â XML ÄÚµåÀÇ ÇÏÀÌ¶óÀÌÆ®ÀÌ´Ù. ÀÌ°ÍÀº ´Ù¸¥ ¹®¼¸¦ Æ÷ÇÔ(include)ÇÏ°í ÇØ¼®(parse)ÇÏ±â À§ÇØ external entity reference handler¸¦ ¾î¶»°Ô »ç¿ëÏ´Â°¡ °¡¿©ÁÖ°í, ¾î¶»°Ô PI(processing instruction)°¡ ¼öÇàµÇ°í, PI°¡ ´ã°í ÀÖ´Â ÄÚµå ÀÇ "½Å·Ú(trust)"¸¦ °áÁ¤Ï´Â °¡¿¡ ´ëÇÑ ¹æ¹ýÀ» °¡¿©ÁØ´Ù.

ÀÌ ¿¹Á¦¸¦ »ç¿ëÏ±â À§ÇÑ XML ¹®¼´Â ¾Æ·¡ ¿¹Á¦¿¡¼ Ã£À» ¼ö ÀÖ´Ù. (`xmltest.xml`¿Í `xmltest2.xml`.)

Example 3. External Entity Example

```
$file = "xmltest.xml";
```

```
function trustedFile($file)
{
    // only trust local files owned by ourselves
    if (!eregi("^([a-z]+)://", $file) && fileowner($file) == getmyuid()) {
        return true;
    }
    return false;
}
function startElement($parser, $name, $attribs)
{
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}
function endElement($parser, $name)
{
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}
function characterData($parser, $data)
{
    print "<b>$data</b>";
}
function PIHandler($parser, $target, $data)
{
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // If the parsed document is "trusted", we say it is safe
            // to execute PHP code inside it.  If not, display the code
            // instead.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>", htmlspecialchars($data));
            }
            break;
    }
}
function defaultHandler($parser, $data)
{
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>', htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>', htmlspecialchars($data));
    }
}
function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId)
{
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}
function new_xml_parser($file) {
    global $parser_file;
    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");
```

```
    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}
if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}
print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);
?>
```

Example 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!-- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </sect1>
</chapter>
```

ÀÌ ÆÄÀÏÀº xmltest.xml¿¡ Æ÷ÇÔ(include)µÈ´Ù.

Example 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
 <element attrib="value"/>
 &testEnt;
 <?php print "This is some more PHP code being executed."; ?>
</foo>
```

---

xml_parser_create

xml_parser_create - - XML parser¸¦ ¸¸µç´Ù.

Description

int xml_parser_create(string [encoding]);

encoding (optional)
Which character encoding the parser should use. The following character encodings are supported:
`ISO-8859-1` (default)
`US-ASCII`
`UTF-8`

This function creates an XML parser and returns a handle for use by other XML functions. Returns `false` on failure.

xml_set_element_handler

xml_set_element_handler - - ½ÃÀÛ element handlers¿Í ³¡ element handlers¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_element_handler`(int parser, string startElementHandler, string endElementHandler);

Sets the element handler functions for the XML parser *parser*. *startElementHandler* and *endElementHandler* are strings containing the names of functions that must exist when **xml_parse()** is called for *parser*.

The function named by *startElementHandler* must accept three parameters:

startElementHandler(int parser, string name, string attribs);

parser
The first parameter, *parser*, is a reference to the XML parser calling the handler.
name
The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.
attribs
The third parameter, *attribs*, contains an associative array with the element's attributes (if any). The keys of this array are the attribute names, the values are the attribute values. Attribute names are case-folded on the same criteria as element names. Attribute values are *not* case-folded.
The original order of the attributes can be retrieved by walking through *attribs* the normal way, using **each()**. The first key in the array was the first attribute, and so on.

The function named by *endElementHandler* must accept two parameters:

endElementHandler(int parser, string name);

parser
The first parameter, *parser*, is a reference to the XML parser calling the handler.
name
The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handlers are set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_set_character_data_handler

xml_set_character_data_handler - - character data handler¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_character_data_handler`(int parser, string handler);

Sets the character data handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept two parameters:

handler(int parser, string data);

parser
The first parameter, *parser*, is a reference to the XML parser calling the handler.
data
The second parameter, *data*, contains the character data as a string.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_set_processing_instruction_handler

xml_set_processing_instruction_handler - - processing instruction (PI) handler¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_processing_instruction_handler`(int parser, string handler);

Sets the processing instruction (PI) handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

A processing instruction has the following format:

`<?target data?>`

You can put PHP code into such a tag, but be aware of one limitation: in an XML PI, the PI end tag (`?>`) can not be quoted, so this character sequence should not appear in the PHP code you embed with PIs in XML documents. If it does, the rest of the PHP code, as well as the "real" PI end tag, will be treated as character data.

The function named by *handler* must accept three parameters:

handler(int parser, string target, string data);

parser
: The first parameter, *parser*, is a reference to the XML parser calling the handler.

target
: The second parameter, *target*, contains the PI target.

data
: The third parameter, *data*, contains the PI data.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_set_default_handler

xml_set_default_handler - - ±â°» handler¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_default_handler`(int parser, string handler);

Sets the default handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept two parameters:

handler(int parser, string data);

parser
: The first parameter, *parser*, is a reference to the XML parser calling the handler.

data
: The second parameter, *data*, contains the character data. This may be the XML declaration, document type declaration, entities or other data for which no other handler exists.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_set_unparsed_entity_decl_handler

xml_set_unparsed_entity_decl_handler - - unparsed entity declaration handler¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_unparsed_entity_decl_handler`(int parser, string handler);

Sets the unparsed entity declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

This handler will be called if the XML parser encounters an external entity declaration with an NDATA declaration, like the following:

`<!ENTITY` *name* {*publicId* | *systemId*} `NDATA` *notationName*`>`

See section 4.2.2 of the XML 1.0 spec for the definition of notation declared external entities.

The function named by *handler* must accept six parameters:

handler(int parser, string entityName, string base, string systemId, string publicId, string notationName);

parser
: The first parameter, *parser*, is a reference to the XML parser calling the handler.

entityName
: The name of the entity that is about to be defined.

base
: This is the base for resolving the system identifier (*systemId*) of the external entity. Currently this parameter will always be set to an empty string.

systemId
: System identifier for the external entity.

publicId
: Public identifier for the external entity.

notationName
: Name of the notation of this entity (see **xml_set_notation_decl_handler()**).

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_set_notation_decl_handler

xml_set_notation_decl_handler - - notation declaration handler¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_notation_decl_handler`(int parser, string handler);

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

A notation declaration is part of the document's DTD and has the following format:

`<!NOTATION` *name* {*systemId* | *publicId*}`>`

See section 4.7 of the XML 1.0 spec for the definition of notation declarations.

The function named by *handler* must accept five parameters:

handler(int parser, string notationName, string base, string systemId, string publicId);

parser
: The first parameter, *parser*, is a reference to the XML parser calling the handler.

notationName
: This is the notation's *name*, as per the notation format described above.

base
: This is the base for resolving the system identifier (*systemId*) of the notation declaration. Currently this parameter will always be set to an empty string.

systemId
: System identifier of the external notation declaration.

publicId
: Public identifier of the external notation declaration.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_set_external_entity_ref_handler

xml_set_external_entity_ref_handler - - external entity reference handler¸¦ ¼³Á¤ÇÑ´Ù.

Description

`int xml_set_external_entity_ref_handler`(int parser, string handler);

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept five parameters, and should return an integer value. If the value returned from the handler is false (which it will be if no value is returned), the XML parser will stop parsing and **xml_get_error_code()** will return XML_ERROR_EXTERNAL_ENTITY_HANDLING.

`int` *handler*(int parser, string openEntityNames, string base, string systemId, string publicId);

parser
: The first parameter, *parser*, is a reference to the XML parser calling the handler.

openEntityNames
: The second parameter, *openEntityNames*, is a space-separated list of the names of the entities that are open for the parse of this entity (including the name of the referenced entity).

base
: This is the base for resolving the system identifier (*systemid*) of the external entity. Currently this parameter will always be set to an empty string.

systemId
: The fourth parameter, *systemId*, is the system identifier as specified in the entity declaration.

publicId
: The fifth parameter, *publicId*, is the public identifier as specified in the entity declaration, or an empty string if none was specified; the whitespace in the public identifier will have been normalized as required by the XML spec.

If a handler function is set to an empty string, or `false`, the handler in question is disabled.

True is returned if the handler is set up, false if *parser* is not a parser.

There is currently no support for object/method handlers.

xml_parse

xml_parse - - XML documentÀÇ ÇØ¼®(parsing)À» ½ÃÀÛÇÑ´Ù.

Description

`int xml_parse`(int parser, string data, int [isFinal]);

parser
: A reference to the XML parser to use.

data
: Chunk of data to parse. A document may be parsed piece-wise by calling **xml_parse()** several times with new data, as long as the *isFinal* parameter is set and true when the last data is parsed.

isFinal (optional)
: If set and true, *data* is the last piece of data sent in this parse.

When the XML document is parsed, the handlers for the configured events are called as many times as necessary, after which this function returns true or false.

True is returned if the parse was successful, false if it was not successful, or if *parser* does not refer to a valid parser. For unsuccessful parses, error information can be retrieved with **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()** and **xml_get_current_byte_index()**.

xml_get_error_code

xml_get_error_code - - XML parser error code¸¦ ¾ò´Â´Ù.

Description

```
int xml_get_error_code(int parser);
```

parser
A reference to the XML parser to get error code from.

This function returns false if *parser* does not refer to a valid parser, or else it returns one of the error codes listed in the error codes section.

xml_error_string

xml_error_string - - XML parser error ¹®ÀÚ¿-À» ¾ò´Â´Ù.

Description

```
string xml_error_string(int code);
```

code
An error code from **xml_get_error_code()**.

Returns a string with a textual description of the error code *code*, or false if no description was found.

xml_get_current_line_number

xml_get_current_line_number - - XML parserÀÇ ÇöÀç line number¸¦ ¾ò´Â´Ù.

Description

```
int xml_get_current_line_number(int parser);
```

parser
A reference to the XML parser to get line number from.

This function returns false if *parser* does not refer to a valid parser, or else it returns which line the parser is currently at in its data buffer.

xml_get_current_column_number

xml_get_current_column_number - - XML parserÀÇ ÇöÀç column number¸¦ ¾ò´Â´Ù.

Description

```
int xml_get_current_column_number(int parser);
```

parser
A reference to the XML parser to get column number from.

This function returns false if *parser* does not refer to a valid parser, or else it returns which column on the current line (as given by **xml_get_current_line_number()**) the parser is currently at.

xml_get_current_byte_index

xml_get_current_byte_index - - XML parserÀÇ ÇöÀç byte index¸¦ ¾ò´Â´Ù.

Description

```
int xml_get_current_byte_index(int parser);
```

parser
A reference to the XML parser to get byte index from.

This function returns false if *parser* does not refer to a valid parser, or else it returns which byte index the parser is currently at in its data buffer (starting at 0).

xml_parser_free

xml_parser_free - - XML parser¸¦ ÇØÁ¦(free)ÇÑ´Ù.

Description

```
string xml_parser_free(int parser);
```

parser
A reference to the XML parser to free.

This function returns false if *parser* does not refer to a valid parser, or else it frees the parser and returns true.

xml_parser_set_option

xml_parser_set_option - - XML parserÀÇ ¿É¼ÇÀ» ¼³Á¤ÇÑ´Ù.

Description

```
int xml_parser_set_option(int parser, int option, mixed value);
```

parser
A reference to the XML parser to set an option in.
option
Which option to set. See below.
value
The option's new value.

This function returns false if *parser* does not refer to a valid parser, or if the option could not be set. Else the option is set and true is returned.

The following options are available:

Table 1. XML parser options

Option constant	Data type	Description
XML_OPTION_CASE_FOLDING	integer	Controls whether case-folding is enabled for this XML parser. Enabled by default.
XML_OPTION_TARGET_ENCODING	string	Sets which target encoding to use in this XML parser. By default, it is set to the same as the source encoding used by **xml_parser_create()**. Supported target encodings are ISO-8859-1, US-ASCII and UTF-8.

xml_parser_get_option

xml_parser_get_option - - XML parserÀÇ ¿É¼ÇÀ» ±¸ÇÑ´Ù.

Description

```
mixed xml_parser_get_option(int parser, int option);
```

parser
A reference to the XML parser to get an option from.
option
Which option to fetch. See **xml_parser_set_option()** for a list of options.

This function returns false if *parser* does not refer to a valid parser, or if the option could not be set. Else the option's value is returned.

See **xml_parser_set_option()** for the list of options.

utf8_decode

utf8_decode - - UTF-8À¸·Î encodeµÈ ¹®ÀÚ¿À» ISO-8859-1·Î ¹Ù²Ù±âÇÑ´Ù.

Description

```
string utf8_decode(string data);
```

This function decodes *data*, assumed to be UTF-8 encoded, to ISO-8859-1.

See **utf8_encode()** for an explaination of UTF-8 encoding.

utf8_encode

utf8_encode - - ISO - 8859 - 1·Î encodeµÈ ¹®ÀÚ¿-À» UTF - 8À¸·Î °¯È¯ÇÑ´Ù.

Description

string utf8_encode(string data);

This function encodes the string *data* to UTF-8, and returns the encoded version. UTF-8 is a standard mechanism used by Unicodefor encoding *wide character* values into a byte stream. UTF-8 is transparent to plain ASCII characters, is self - synchronized (meaning it is possible for a program to figure out where in the bytestream characters start) and can be used with normal string comparison functions for sorting and such. PHP encodes UTF-8 characters in up to four bytes, like this:

Table 1. UTF - 8 encoding

bytes	bits	representation
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Each *b* represents a bit that can be used to store character data.

III. ºÎ·Ï (Appendixes)

Table of Contents

Appendix A. Migrating from PHP/FI 2.0 to PHP 3.0

(¿ªÀÚÁÖ: ÀÌºÎºÐÀº °°·Î ÀÌ¿ëµÇÁö ¾ÊÀ» °ÍÀ¸·Î »ý°¢µÇ¹Ç·Î ¹ø¿ªÀ» »ý·«ÇÑ´Ù.)

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.

Start/end tags

The first thing you probably will notice is that PHP's start and end tags have changed. The old `<? >` form has been replaced by three new possible forms:
Example 0 - 1. Migration: old start/end tags

```
<? echo "This is PHP/FI 2.0 code.\n"; >
```

As of version 2.0, PHP/FI also supports this variation:
Example 0 - 2. Migration: first new start/end tags

```
<? echo "This is PHP 3.0 code!\n"; ?>
```

Notice that the end tag now consists of a question mark and a greater-than character instead of just greater-than. However, if you plan on using XML on your server, you will get problems with the first new variant, because PHP may try to execute the XML markup in XML documents as PHP code. Because of this, the following variation was introduced:
Example 0 - 3. Migration: second new start/end tags

```
<?php echo "This is PHP 3.0 code!\n"; ?>
```

Some people have had problems with editors that don't understand the processing instruction tags at all. Microsoft FrontPage is one such editor, and as a workaround for these, the following variation was introduced as well:
Example 0 - 4. Migration: third new start/end tags

```
<script language="php">
  echo "This is PHP 3.0 code!\n";
</script>
```

if..endif syntax

The `alternative' way to write if/elseif/else statements, using if(); elseif(); else; endif; cannot be efficiently implemented without adding a large amount of complexity to the 3.0 parser. Because of this, the syntax has been changed:
Example 0 - 5. Migration: old if..endif syntax

```
if ($foo);
```

```
    echo "yep\n";
elseif ($bar);
    echo "almost\n";
else;
    echo "nope\n";
endif;
```

Example 0 - 6. Migration: new if..endif syntax

```
if ($foo):
    echo "yep\n";
elseif ($bar):
    echo "almost\n";
else:
    echo "nope\n";
endif;
```

Notice that the semicolons have been replaced by colons in all statements but the one terminating the expression (endif).

---

while syntax

Just like with if..endif, the syntax of while..endwhile has changed as well:
Example 0 - 7. Migration: old while..endwhile syntax

```
while ($more_to_come);
    ...
endwhile;
```

Example 0 - 8. Migration: new while..endwhile syntax

```
while ($more_to_come):
    ...
endwhile;
```

Warning
If you use the old while..endwhile syntax in PHP 3.0, you will get a never - ending loop.

Expression types

PHP/FI 2.0 used the left side of expressions to determine what type the result should be. PHP 3.0 takes both sides into account when determining result types, and this may cause 2.0 scripts to behave unexpectedly in 3.0.

Consider this example:

```
$a[0]=5;
$a[1]=7;
$key = key($a);
while ("" != $key) {
    echo "$keyn";
    next($a);
}
```

In PHP/FI 2.0, this would display both of $a's indices. In PHP 3.0, it wouldn't display anything. The reason is that in PHP 2.0, because the left argument's type was string, a string comparison was made, and indeed "" does not equal "0", and the loop went through. In PHP 3.0, when a string is compared with an integer, an integer comparison is made (the string is converted to an integer). This results in comparing atoi("") which is 0, and variablelist which is also 0, and since 0==0, the loop doesn't go through even once.

The fix for this is simple. Replace the while statement with:

```
while ((string)$key != "") {
```

Error messages have changed

PHP 3.0's error messages are usually more accurate than 2.0's were, but you no longer get to see the code fragment causing the error. You will be supplied with a file name and a line number for the error, though.

Short - circuited boolean evaluation

In PHP 3.0 boolean evaluation is short - circuited. This means that in an expression like (1 || test_me()), the

function **test_me()** would not be executed since nothing can change the result of the expression after the `1`.

This is a minor compatibility issue, but may cause unexpected side - effects.

Function true/false return values

Most internal functions have been rewritten so they return TRUE when successful and FALSE when failing, as opposed to 0 and - 1 in PHP/FI 2.0, respectively. The new behaviour allows for more logical code, like `$fp = fopen("/your/file") or fail("darn!");`. Because PHP/FI 2.0 had no clear rules for what functions should return when they failed, most such scripts will probably have to be checked manually after using the 2.0 to 3.0 convertor.

Example 0 - 9. Migration from 2.0: return values, old code

```
$fp = fopen($file, "r");
if ($fp == -1);
    echo("Could not open $file for reading<br>\n");
endif;
```

Example 0 - 10. Migration from 2.0: return values, new code

```
$fp = @fopen($file, "r") or print("Could not open $file for reading<br>\n");
```

Other incompatibilities

The PHP 3.0 Apache module no longer supports Apache versions prior to 1.2. Apache 1.2 or later is required.

echo() no longer supports a format string. Use the **printf()** function instead.

In PHP/FI 2.0, an implementation side - effect caused `$foo[0]` to have the same effect as `$foo`. This is not true for PHP 3.0.

Reading arrays with `$array[]` is no longer supported

That is, you cannot traverse an array by having a loop that does `$data = $array[]`. Use **current()** and **next()** instead.

Also, `$array1[] = $array2` does not append the values of `$array2` to `$array1`, but appends `$array2` as the last entry of `$array1`. See also multidimensional array support.

`"+"` is no longer overloaded as a concatenation operator for strings, instead it converts it's arguments to numbers and performs numeric addition. Use `"."` instead.

Example 0 - 11. Migration from 2.0: concatenation for strings

```
echo "1" + "1";
```

In PHP 2.0 this would echo 11, in PHP 3.0 it would echo 2. Instead use:

```
echo "1"."1";
$a = 1;
$b = 1;
echo $a + $b;
```

This would echo 2 in both PHP 2.0 and 3.0.

```
$a = 1;
$b = 1;
echo $a.$b;
```

This will echo 11 in PHP 3.0.

Appendix B. PHP development

Adding functions to PHP3

Function Prototype

¸ðµç ÇÔ¼ö´Â ´ÙÀ½°ú °°Àº ¸ð¾çÀ» ÃëÇÑ´Ù. :

```
void php3_foo(INTERNAL_FUNCTION_PARAMETERS) {

}
```

°ñ·Ï ÇÔ¼ö°¡ ¾î¶² ÀÎ¼ö(argument)µµ °¡Áö°í ÀÖÁö ¾Ê¾Æµµ, ÀÌ·± ¹æ½ÄÀ¸·Î È£ÃâµÈ´Ù.

Function Arguments

ÀÎ¼ö(Argument)´Â ÇÑ»ó pval typeÀÌ´Ù. ÀÌ typeÀº argumentÀÇ actual typeÀ» °¡Áø unionÀ» Æ÷ÇÔÇÏ°í ÀÖ´Ù. µû¶ó¼, ¸¸¾à ÇÔ¼ö°¡ µÎ °³ÀÇ argument¸¦ °¡Áø´é, ÇÔ¼öÀÇ ¸Ç Ã³À½¿¡¼ ´ÙÀ½°ú °°Àº µ¿ÀÛÀ» ÃëÇÑ´Ù. :
Example 0 - 1. Fetching function arguments

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

NOTE: argument´Â by value¿Í by referenceÀÇ ¹æ½ÄÁß¿¡ ÇÏ³ª·Î Àü´ÞµÈ´Ù. µÎ ¹æ½Ä ¸ðµÎ getParameters¿¡ &(pval *)À» Àü´ÞÇÒ ÇÊ¿ä°¡ ÀÖ´Ù. ¸¸¿ª ¿©·¯°ÐÀÌ n¹øÂ° parameter°¡ by reference·Î Àü´ÞµÇ¾ú´Â°¡¸¦ °Ë»çÇÏ·Á¸é, ParameterPassedByReference(ht,n) ÇÔ¼ö¸¦ »ç¿ëÇÑ´Ù. ÀÌ ÇÔ¼ö´Â 1ÀÌ³ª 0À» ¹ÝÈ¯ÇÑ´Ù.

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:
Example 0 - 2. Variable function arguments

```
pval *arg1, *arg2, *arg3;
int arg_count = ARG_COUNT(ht);
if (arg_count < 2 || arg_count > 3 ||
    getParameters(ht,arg_count,&arg1,&arg2,&arg3)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

Using the Function Arguments

°³°° argumentÀÇ typeÀº pval type ÇÊµå¿¡ ÀúÀåµÈ´Ù. ÀÌ typeÀº ´ÙÀ½Áß ÇÑ °³´Ù. :

Table 0 - 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:

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

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

ÇÔ¼ö°¡ ÇÊ¿äÇÑ ¸ðµç ¸Þ¸ð¸®´Â emalloc()ÀÌ³ª estrdup()À» »ç¿ëÇØ¼ È®º¸ÇÑ´Ù. ÀÌ ÇÔ¼öµéÀº ÀÏ¹ÝÀûÀÎ malloc(), strdup() °ú ºñ½ÁÇÏ°Ô ´À²¸Áö´Â ÀßºóÀûÀÎ ¸Þ¸ð¸® °ü¸® ÇÔ¼öÀÌ´Ù. ÀÌ ¸Þ¸ð¸®´Â efree()ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ÇØÁ¦(free)µÈ´Ù.

ÇÁ·Î±×·¥¿¡¼´Â ´ÙÀ½°ú °°Àº µÎ Á¾·ùÀÇ Á¾·ùÀÇ ¸Þ¸ð¸®°¡ »ç¿ëµÈ´Ù. : º¯¼ö¿¡¼ »ç¿ëÇÏ¿© ÆÄ¼¿¡ µ¹·ÁÄö´Â ¸Þ¸ð¸®¿Í, ¿©·¯ºÐÀÇ ³»ºÎ ÇÔ¼ö(internal function)¿¡¼ ÀÓ½Å ÀúÅå °ø°£À¸·Î ÇÊ¿äÇÑ ¸Þ¸ð¸®. ¸¸à ¿©·¯ºÐÀÌ ¹®ÀÚ¿À» ÆÄ¼¿¡ µ¹·ÁÀÙ º¯¼ö¿¡ ÀúÅåÇÏ·Á ÇÑ´Ù¸é, ÀÌ º¯¼ö´Â ¿ì¼± emalloc()ÀÌ³ª estrdup()ÀÇ ÇÔ¼ö¸¦ »ç¿ëÇÏ¿© ¸Þ¸ð¸®ÀÇ °ø°£À» È®º¸ÇÒ ÇÊ¿ä° ÀÖ´Ù. ÀÌ ¸Þ¸ð¸®´Â Àý´ë ¿©·¯ºÐÀÌ Ç®¾îÁÙ ¼ø´Ù. ´ÜÁö, ÈÄ¿¡ °°Àº ÇÔ¼ö³»¿¡¼ ¿ø·¡ÀÇ ´ëÀÛµÈ °ªÀ» ¹Ù²Ù¾î ÁÙ ¼ö ÀÖÀ» »ÓÀÌ´Ù.

¿©·¯ºÐÀÌ ÇÔ¼ö³ª ¶óÀÌºê·¯¸®¿¡¼ ÇÊ¿äÇÑ ¾îî¶°ÇÑ ÀÓ½Å/¿µ±¸ÀûÀÎ ¸Þ¸ð¸®¿¡ ´ëÇØ¼µµ, ¿©·¯ºÐÀº emalloc()°ú estrdup(), efree()ÀÇ 3°³ÀÇ ÇÔ¼ö¸¦ »ç¿ëÇØ¾ß ÇÑ´Ù. ÀÌ ÇÔ¼öµéÀº ¹Ýµå½Å ±×°Í¿¡ ´ëÀÀµÇ´Â ÇÔ¼öµé°ú ÇÔ²² ÀÛµ¿ÇØ¾ß ÇÑ´Ù. emalloc() ÀÌ³ª estrdup()À» »ç¿ëÇÑ ¸Þ¸ð¸®¿¡´Â, ÀÌ ¸Þ¸ð¸®°¡ ÇÁ·Î±×·¥ÉÍ ¶§±îÁö ÇÊ¿äÇÑ °ÍÀÌ ¾Æ´Ñ ÇÑ, ¹Ýµå½Ã efree()°¡ ÇÊ¿äÇÏ´Ù. ¸¸à ±×·¸Ô ÇÏÁö ¾ÊÀ¸¸é ¸Þ¸ð¸® ´©¼³(leak)ÀÌ ¹ß»ýÇÑ´Ù. "±× ÇÔ¼ö°¡ ±×°Í¿¡ ´ëÀÀµÇ´Â ÇÔ¼ö¿Í ÇÔ²² ÀÛµ¿ÇØ¾ß ÇÑ´Ù" ´Â °ÍÀÇ ÀÇ¹Ì´Â, emalloc()ÀÌ³ª estrdup()¾øÀÌ efree()¸¦ »ç¿ëÇÑ´Ù¸é ¿©·¯ºÐÀº segment fault ¿À·ù¸¦ ³³°Ô µÉ °ÍÀÌ´Ù. ±×·¯¹Ç·Î ÇÊ¿ä¾ø´Â ¸Þ¸ð¸®¿¡ ´ëÇÑ ÇØÁ¦¸´Â Ãæºê ÑÇÑ ÁÖÀÇ¸¦ ÇÒ °ÍÀÌ´Ù.

PHP3À» "-DDEBUG"¸¦ ÁÖ°í ÄÄÆÄÀÏÇÏ¿´´Ù¸é, PHP3´Â emalloc()ÀÌ³ª estrdup()¿¡ ÀÇÇØ È®º¸µÈ ¸ðµç ¸Þ¸ð¸® list¸¦ Ç¥½ÃÇÒ °ÍÀÌ´Ù. ±×·¯³ª, efree()ÇÔ¼ö¸¦ »ç¿ëÇØµµ ÇØÁ¦µÇÁö ¾Ê°í, ÇØ´ç ÇÁ·Î±×·¥ÀÌ Á¾·áµÉ ¶§ ÇØÁ¦µÈ´Ù.

Setting Variables in the Symbol Table

symbol table¿¡ º¯¼ö¸¦ ¼³Á¤ÇÏ±â ½±°ÔÇÏ±â À§ÇØ ´ÙÀ½°ú °°Àº ¸î °³ÀÇ macro°¡ ÀÖ´Ù. :

SET_VAR_STRING(name,value) [1]

SET_VAR_DOUBLE(name,value)

SET_VAR_LONG(name,value)

[1]

PHP 3.0ÀÇ Symbol tablesÀº hash tableÀÇ ÇüÅÂ·Î µÇ¾î ÀÖ´Ù. ¾ðÁ¦³ª &symbol_tableÀº 'main' symbol tableÀ» °¡¸®Å°°í ÀÖ°í, active_symbol_tableÀº ÇöÀç activeµÈ symbol tableÀ» °¡¸®Å°°í ÀÖ´Ù. (ÀÌ µÎ°¡Áö ½ÃÀÛ(startup)½Ã´Â °°Áö¸¸, ÇÔ¼ö¾È¿¡¼´Â ´¶óÁø´Ù.)

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.

Example 0 - 3. Checking whether $foo exists in a symbol table

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

Example 0 - 4. Finding a variable's size in a symbol table

```
hash_find(active_symbol_table,"foo",sizeof("foo"),&pvalue);
check(pvalue.type);
```

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:
Example 0 - 5. Initializing a new array

```
pval arr;
if (array_init(&arr) == FAILURE) { failed... };
hash_update(active_symbol_table,"foo",sizeof("foo"),&arr,sizeof(pval),NULL);
```

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:
Example 0 - 6. Adding entries to a new array

```
pval entry;
entry.type = IS_LONG;
entry.value.lval = 5;

/* defines $foo["bar"] = 5 */
hash_update(arr.value.ht,"bar",sizeof("bar"),&entry,sizeof(pval),NULL);
/* defines $foo[7] = 5 */
hash_index_update(arr.value.ht,7,&entry,sizeof(pval),NULL);
/* defines the next free place in $foo[],
 * $foo[8], to be 5 (works like php2)
 */
hash_next_index_insert(arr.value.ht,&entry,sizeof(pval),NULL);
```

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:

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

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

```
add_next_index_long(return_value,long_value);
add_next_index_double(return_value,double_value);
add_next_index_string(return_value,estrdup(string_value));
```

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

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

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

ÇÔ¼ö¿¡¼- ½±°Ô °ªÀ» ¹ÝÈ¯ÇÏ±â À§ÇØ ´ÙÀ½°ú °°Àº ¸î °³ÀÇ ¸ÅÅ©·Î°¡ ÀÖ´Ù. :

¸ðµç RETURN_* ¸ÅÅ©·ÎµéÀº return°ªÀ» ¼³Á¤ÇÏ°í, ÇØ´ç ÇÔ¼ö·Î°ÎÅÍ °¹±ÍÑ´Ù. :

- 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)

¸ðµç RETVAL_* ¸ÅÅ©·ÎµéÀº return°ªÀ» ¼³Á¤ÇÏÁö¸¸, ÇØ´ç ÇÔ¼ö·Î°ÎÅÍ °¹±ÍÇÏÁö´Â ¾Ê´Â´Ù. :

- 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

ÇÔ¼ö´Â object³ª array°°Àº complex dataµµ ¹ÝÈ¯ÇÒ ¼ö ÀÖ´Ù.

Returning an object:

Call object_init(return_value).

Fill it up with values. The functions available for this purpose are listed below.

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:

```
add_method( return_value, function_name, function_ptr );
```

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:

Call array_init(return_value).

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)

add_assoc_stringl(return_value,key,str,length) 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 3.0À° ¿©·¯ ÀÚ¿øÀÇ °¡Áö ÇüÅÂ¸¦ ´Ù·ç´Â Ç¥ÁØÀûÀÎ ¹æ¹ýÀ» °¡Áö°í ÀÖ´Ù. ÀÌ°ÍÀº PHP 2.0ÀÇ ¸ðµç Áö¿ªÀûÀÎ linked list¸¦ ´ëÃ¼ÇÑ´Ù.

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:
Example 0 - 7. Adding a new resource

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

Example 0 - 8. Using an existing resource

```
pval *resource_id;
RESOURCE *resource;
int type;
convert_to_long(resource_id);
resource = php3_list_find(resource_id->value.lval, &type);
if (type != LE_RESOURCE_TYPE) {
        php3_error(E_WARNING,"resource index %d has the wrong type",resource_id->value.lval);
        RETURN_FALSE;
}
/* ...use resource... */
```

Example 0 - 9. Deleting an existing resource

```
pval *resource_id;
RESOURCE *resource;
int type;
convert_to_long(resource_id);
php3_list_delete(resource_id->value.lval);
```

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 3.0À° ¿µ±¸ÀûÀÎ ÀÚ¿ø(persistent resources; i.e., resources that are kept in between hits)À» ÀúÀåÇÏ´Â Ç¥ÁØÀûÀÎ ¹æ¹ýÀ» °¡Áö°í ÀÖ´Ù. ÀÌ ¹æ¹ýÀº MySQL ¸ðµâ¿¡¼ Â³À½ç¿ëµÇ¾ú°í, mSQLµµ ÀÌ°ÍÀ» µû¸£°í ÀÖ´Ù. µû¶ó¼ persistent resource¸¦ ¾î¶»°Ô »ç¿ëÇÏ´Â°¡¿¡ ´ëÇØ ¾Ë°í ½Í´Ù¸é ¿ì¼± mysql.c¸¦ ÀÐ¾î°¡µµ ÏÇÏÀÚ. ÁÖÀÇÇØ¼ º¼ ÇÔ¼öµéÀº ´ÙÀ°°ú °°´Ù. :

php3_mysql_do_connect
php3_mysql_connect()
php3_mysql_pconnect()

persistence ¸ðµâÀÇ ÀÏ¹ÝÀûÀÎ idea´Â ´ÙÀ½°ú °°´Ù. :

Code all of your module to work with the regular resource list mentioned in section (9).

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

¸¹À° PHP3ÀÇ ±â´É(feature)µéÀÌ ½ÇÇàÁß¿¡ ¼³Á¤ °¡´ÉÇÏ´Ù. ÀÌ ¼³Á¤ Áö½ÃÀÚ(configuration directives)´Â php3.ini¿¡ ¼³Á¤µÇ°Å³ª, Apache ¸ðµâÀÇ °æ¿ì .conf ÆÄÀÏ¿¡ ¼³Á¤°¡´ÉÏµµ·Ï µÇ¾î ÀÖ´Ù. Apache .conf ÆÄÀÏ¿¡ ¼³Á¤ÇÏ´Â °ÍÀÇ ÀåÁ¡À° µð·°Åä¸®°°·Î ¼³Á¤Å» ´Ù¸£°Ô ÇÒ ¼ö ÀÖ´Ù´Â Á¡ÀÌ´Ù. ÀÌ°ÍÀº ¿¹¸¦µé¾î ¾î¶² µð·°Åä¸®°¡ ´Ù¸¥ µð·°Åä¸®¸¦ °¡Áö°í ÀÖ¾îµµ, ÇØ´ç ÇÏ´Â ÇÑ µð·°Åä¸®¿¡¸¸ safemodeexecdir ¼³Á¤Å» ÇÒ ¼ö ÀÖ´Ù´Â °ÍÀÌ´Ù. ÀÌ °³°° ¼³Á¤ ±â´ÉÀº ¼-¹ö¡ multiple virtual hosts¸¦ Áö¿øÇÒ ¶§ ¹«Ã´ À¯¿ëÏ´Ù.

The steps required to add a new directive:

Add directive to php3_ini_structure struct in mod_php3.h.

In main.c, edit the php3_module_startup function and add the appropriate cfg_get_string() or cfg_get_long() call.

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.

In either php3take1handler() or php3flaghandler() add the appropriate entry for your directive.

In the configuration section of the _php3_info() function in functions/info.c you need to add your new directive.

And last, you of course have to use your new directive somewhere. It will be addressable as php3_ini.directive.

Calling User Functions

³»°Î ÇÔ¼ö(internal function)¿¡¼- »ç¿ëÀÚ ÇÔ¼ö¸¦ °Î¸£·Á¸é, **call_user_function()** ÇÔ¼ö¸¦ »ç¿ëÇÏ¿©¾ß ÇÑ´Ù.

call_user_function() returns SUCCESS on success, and FAILURE in case the function cannot be found. You should check that return value! If it returns SUCCESS, you are responsible for destroying the retval pval yourself (or return it as the return value of your function). If it returns FAILURE, the value of retval is undefined, and you mustn't touch it.

All internal functions that call user functions *must* be reentrant. Among other things, this means they must not use globals or static variables.

call_user_function() takes six arguments:

HashTable *function_table

This is the hash table in which the function is to be looked up.

---

pval *object

This is a pointer to an object on which the function is invoked. This should be NULL if a global function is called. If it's not NULL (i.e. it points to an object), the function_table argument is ignored, and instead taken from the object's hash. The object *may* be modified by the function that is invoked on it (that function will have access to it via $this). If for some reason you don't want that to happen, send a copy of the object instead.

---

pval *function_name

The name of the function to call. Must be a pval of type IS_STRING with function_name.str.val and function_name.str.len set to the appropriate values. The function_name is modified by call_user_function() - it's converted to lowercase. If you need to preserve the case, send a copy of the function name instead.

---

pval *retval

A pointer to a pval structure, into which the return value of the invoked function is saved. The structure must be previously allocated - **call_user_function()** does NOT allocate it by itself.

---

int param_count

The number of parameters being passed to the function.

---

pval *params[]

An array of pointers to values that will be passed as arguments to the function, the first argument being in offset 0, the second in offset 1, etc. The array is an array of pointers to pval's; The pointers are sent as-is to the function, which means if the function modifies its arguments, the original values are changed (passing by reference). If you don't want that behavior, pass a copy instead.

Reporting Errors

³»°Î ÇÔ¼ö¿¡¼- ¿¡·¯¸¦ reportÇÒ ¶§´Â **php3_error()**ÇÔ¼ö¸¦ »ç¿ëÇÏ´Â °ÍÀÌ ÁÁ´Ù. ÀÌ ÇÔ¼ö´Â ÃÖ¼Ò µÎ°³ÀÇ ÀÎ¼ö¸¦ °¡Áö°í È£ÃâµÈ´Ù. Ã¹¹øÂ°´Â ¿¡·¯ÀÇ levelÀÌ°í, ´Ù¸¥ ÇÏ³ª´Â ¿¡·¯ ¸Þ¼¼Áö¸¦ À§ÇÑ format string(**printf()**¿¡¼- »ç¿ëµÇ´Â °Í°ú °°À° ÇüÅÂ) ÀÌ´Ù. ±×¸®°í ±× ³ª¸ÓÁö´Â ÁÖ¾îÁø format stringÀÇ parameter°¡ µÈ´Ù. ¿¡·¯ levelÀº ´ÙÀ½°ú °°´Ù. :

E_NOTICE

Notice´Â ±âº»ÀûÀ¸·Î ´Â Ãâ·ÂµÇÁö ¾Ê´Â´Ù. ÀÌ°ÍÀº ½ºÅ©¸³Æ®°¡ ¹°°¡ ¿¡·¯¸¦ °¨ÁöÇßÁö¸¸, ±×°ÍÀÌ Á¤»óÀûÀÎ »óÈ²¿¡¼- ¹ß»ýÇÏ´Â °ÍÀÌ¶ó´Â ÀÇ¹ÌÀÌ´Ù. ¿¹¸¦µé¾î ¼³Á¤µÇÁö ¾ÊÀº °¯¼ö¸¦ »ç¿ëÇÏ·Á ÇßºÃ³ª, Á¸ÀçÇÏÁö ¾Ê´Â ÆÄÀÏ¿¡ **stat()** ÇÔ¼ö¸¦ È£ÃâÇÏ´Â °Í µîÀÌ´Ù.

E_WARNING

WarningÀº ±âº»ÀûÀ¸·Î Ãâ·ÂµÈ´Ù. ±×·¯³ª, ½ºÅ©¸³Æ®ÀÇ ½ÇÇàÀ» ¸ØÃßÁö´Â ¾Ê´Â´Ù. ÀÌ°ÍÀº È£ÃâÀÌ ¿Ï·áµÇ±â Àü¿¡ ½ºÅ©¸³Æ®¿¡ ÀÇÇØ ÀâÇôÁ®¾ß ÇÏ´Â ¹®Á¦¸¦ °¡¸®Å²´Ù. ¿¹¸¦µé¾î ÀßºøÈ regular expressionÀ¸·Î **ereg()**¸¦ È£ÃâÇÏ´Â °Í µîÀÌ´Ù.

E_ERROR

Errorµµ ±â°»ÀûÀ¸·Î Ãâ·ÂµÈ´Ù. ±×¸®®°í, ÇÔ¼öÀÇ ½ÇÇàÀÌ ¿Ï·áµÈ Á÷ÈÄ, ½°Å©¸³Æ®ÀÇ ½ÇÇàÀ» ¸ØÃá´Ù. ÀÌ°ÍÀº memory allocation ¹®Á¦°°Àº °¹±¸°¡ °Ò°¡´ÉÇÑ ¿¡·¯¸¦ ÀÇ¹ÌÇÑ´Ù.

E_PARSE

Parse error´Â ¿ÀÁ÷ Parser¸¸ÀÌ »ý¼°°¡´ÉÇÏ´Ù. code°¡ listµÈ´Ù.

E_CORE_ERROR

PHPÀÇ core¿¡ÀÇÇØ »ý¼°µÈ´Ù´Â Á¡À» Á¦¿ÜÇÏ°í´Â E_ERROR¿Í µ¿ÀÏÇÏ´Ù. ÀÏ¹Ý ÇÔ¼ö·Î´Â ÀÌ ÇüÅÂÀÇ ¿¡·¯¸¦ ¸¸µéÀö ¸øÇÑ´Ù.

E_CORE_WARNING

PHPÀÇ core¿¡ÀÇÇØ »ý¼°µÈ´Ù´Â Á¡À» Á¦¿ÜÇÏ°í´Â E_WARNING¿Í µ¿ÀÏÇÏ´Ù. ÀÏ¹Ý ÇÔ¼ö·Î´Â ÀÌ ÇüÅÂÀÇ ¿¡·¯¸¦ ¸¸µéÀö ¸ø ÇÑ´Ù.

Hitchhiker's guide to PHP internals

Appendix C. The PHP Debugger

Using the Debugger

PHPÀÇ ³»Àå µð¹ö°Å´Â ÆÄ¼ÇÇÏ±â Èûµç ¹ö±×¸¦ Ã£¾Æ³»´Âµ¥ ¸Å¿ì À¯¿ëÇÑ µµ±¸ÀÌ´Ù. µð¹ö°Å´Â PHP°¡ ½ÅÀÛµÉ ¶§¸¶´Ù TCP Æ÷Æ®¿¡ Á¢¼ÓÇÏ¿© µ¿ÀÛÇÑ´Ù. ¿ä±¸¿¡ ´ëÇÑ ¸ðµç ¿¡·¯ ¸Þ¼¼Áö°¡ ÀÌ Á¡¼ÖÀ» ÅëÇØ Àü´ÞµÈ´Ù. This information is intended for "debugging server" that can run inside an IDE or programmable editor (such as Emacs).

debugger ¼³Á¤ ¹æ¹ý:

php3.iniÀÇ (debugger.port)¿¡ µð¹ö°Å¸¦ À§ÇÑ TCP Æ÷Æ®¸¦ ¼³Á¤ÇÏ°í (debugger.enabled)¸¦ ¼³Á¤ÇØ enble½ÅÅ²´Ù.

¾î´À ½Å½°ÅÛ¿¡ TCP listener¸¦ ¼³Á¤ÇÑ Æ÷Æ®·Î ¼°Á¤Ñ´Ù. (¿¹¸¸µé¾î UNIX¿¡¼´Â **socket -l -s 1400** ¶ó°í ½ÇÇàÇÑ´Ù).

code¿¡ "debugger_on(*host*)"¸¦ ½ÇÇàÇÑ´Ù. *host* ´Â TCP listener°¡ µ¹°Í ÀÖ´Â ½Å½°ÅÛÀÇ IP ¹øÈ£³ª host À¸§À¸·Î.

ÀÌÁ¦, ¸ðµç warning°ú notice µîÀÌ listener socket¿¡ Ç¥½ÃµÉ °ÍÀÌ´Ù. ¸¸¾à ¿©·¯°ÐÀÌ ***error_reporting()***¸¦ *»ç¿ëÇØ¼ reportÇÏÁö ¸øÇÏµµ·Ï ÇØµµ ÀÌ ·¹Æ÷Æ®´Â µ¿ÀÛÇÑ´Ù.*

Debugger Protocol

debugger protocolÀº ¶óÀÎ ´ÜÀ§ÀÌ´Ù. °¢ ¶óÀÎÀº *type*À» °¡Áö°í ÀÖ°í, ¿©·¯¶óÀÎÀÌ ÇÏ³ªÀÇ ¸Þ¼¼Áö¸¦ ±¸¼ºÇÑ´Ù. °¢ ¸Þ¼¼Áö´Â start type À¸·Î ½ÀÀÛÇÏ°í, end typeÀ¸·Î ³¡³´Ù. PHP´Â µ¿½Ã¿¡ ¿©·¯ ´Ù¸¥ ¸Þ¼¼ÁöÀÇ ºÐ¸®¸¦ ¾öµµ ÀÖ´Ù.

ÇÑ ÁÙÀº ´ÙÀ½°ú °°Àº formatÀ¸·Î µÇ¾î ÀÖ´Ù.:

ÇÑ ÁÙÀº ´ÙÀ½°ú °°Àº formatÀ¸·Î µÇ¾î ÀÖ´Ù.:

date time host(*pid*) *type*: *message-data*

date
: Date in ISO 8601 format (*yyyy-mm-dd*)

time
: Time including microseconds: *hh:mm:uuuuuu*

host
: script error °¡ »ý±ä DNS name ÀÌ³ª IP address

pid
: ÀÌ ¿¡·¯¸¦ ¸¸µç PHP script processÀÇ *host*ÀÇ PID (process id)

type
: ¶óÀÎÀÇ type. Tells the receiving program about what it should treat the following data as:

Table C - 1. Debugger Line Types

Name	Meaning
start	Tells the receiving program that a debugger message starts here. The contents of *data* will be the type of error message, listed below.
message	The PHP error message.
location	File name and line number where the error occured. The first `location` line will always contain the top-level location. *data* will contain *file*:*line*. There will always be a `location` line after `message` and after every `function`.
frames	Number of frames in the following stack dump. If there are four frames, expect information about four levels of called functions. If no "frames" line is given, the depth should be assumed to be 0 (the error occured at top-level).
function	Name of function where the error occured. Will be repeated once for every level in the function call stack.
end	Tells the receiving program that a debugger message ends here.

data
Line data.

Table C-2. Debugger Error Types

Debugger	PHP Internal
warning	E_WARNING
error	E_ERROR
parse	E_PARSE
notice	E_NOTICE
core-error	E_CORE_ERROR
core-warning	E_CORE_WARNING
unknown	(any other)

Example C-1. Example Debugger Message

```
1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (null):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice
```

ÀÌ ¹®¼´Â ´ÙÀ½°ú °°À° ¸µÅ©¸¦ Æ÷ÇÔÇÕ´Ï´Ù.

¸µÅ© ÀÌ¸§	ÀÎÅÍ³Ý ÁÖ¼Ò
http://w3.to/regina	http://w3.to/regina
regina@officeware.medialab.co.kr	mailto:regina@officeware.medialab.co.kr
DocBook DTD	http://www.ora.com/davenport/
DSSSL	http://www.jclark.com/dsssl/
http://www.php.net	http://www.php.net/
Adabas home page	http://www.adabas.com/
mSQL home page	http://www.hughes.com.au/
MySQL home page	http://www.tcx.se/
FreeODBC home page	http://users.ids.net/~bjepson/freeODBC/
OpenLink Software's home page	http://www.openlinksw.com/
Oracle home page	http://www.oracle.com/
PostgreSQL home page	http://www.postgresql.org/
Solid home page	http://www.solidtech.com/
Sybase home page	http://www.sybase.com/
Velocis home page	http://www.raima.com/
RFC1777	ftp://ftp.isi.edu/in-notes/rfc1777.txt
RFC1778	ftp://ftp.isi.edu/in-notes/rfc1778.txt
expat library	http://www.jclark.com/xml/
Bob Silva	mailto:bob_silva@mail.umesd.k12.or.us
http://www.umesd.k12.or.us/php/win32install.html	http://www.umesd.k12.or.us/php/win32install.html
ChangeLog	http://www.php.net/ChangeLog.php3
FAQ	http://www.php.net/FAQ.php3
mirror	http://www.netvision.net.il/browser-id.php3
tool	http://www.genusa.com/iis/iiscfg.html
http://www.php.net/FAQ.php3	http://www.php.net/FAQ.php3
http://w3.to/regina/FAQ.htm	http://w3.to/regina/FAQ.htm
http://ca.php.net/bugs.php3	http://ca.php.net/bugs.php3
http://www.tryc.on.ca/php3.html	http://www.tryc.on.ca/php3.html
php3-subscribe@lists.php.net	mailto:php3-subscribe@lists.php.net
Qmail	http://www.qmail.org/
CA-96.11	http://www.cert.org/pub/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html
Netscape's Spec	http://www.netscape.com/newsref/std/cookie_spec.html
http://metalab.unc.edu/kevina/aspell/	http://metalab.unc.edu/kevina
http://genealogy.org/~scottlee/cal-overview.html	http://genealogy.org/~scottlee/cal-overview.html
Sleepycat Software's DB2	http://www.sleepycat.com/
GNU database manager	ftp://ftp.gnu.org/pub/gnu/gdbm/
Sleepycat Software's DB2	http://www.sleepycat.com/
http://pobox.com/~djb/cdb.html	http://pobox.com/~djb/cdb.html
http://www.fileproplus.com/	http://www.fileproplus.com/
HTTP 1.1 Specification	http://www.w3.org/Protocols/rfc2068/rfc2068
http://www.netscape.com/newsref/std/cookie_spec.html	http://www.netscape.com/newsref/std/cookie_spec.html
IICM	http://www.iicm.edu/
www.hyperwave.com	http://www.hyperwave.com/
HG-CSP	http://www.hyperwave.de/hg-csp-0xc31b08d4
http://www.boutell.com/gd/	http://www.boutell.com/gd/
http://www.xe.net/iptc/	http://www.xe.net/iptc/
FreeType	http://www.freetype.org/
ftp://ftp.cac.washington.edu/imap/	ftp://ftp.cac.washington.edu/imap/
Netscape	http://developer.netscape.com/tech/directory/
University of Michigan	http://www.umich.edu/~dirsvcs/ldap/index.html
OpenLDAP Project	http://www.openldap.com/
LDAP World	http://elvira.innosoft.com/ldapworld
Netscape Directory SDK	http://developer.netscape.com/tech/directory/
http://www.math.keio.ac.jp/~matumoto/emt.html	http://www.math.keio.ac.jp/~matumoto/emt.html
http://www.scp.syr.edu/~marc/hawk/twister.html	http://www.scp.syr.edu/~marc/hawk/twister.html
ftp://argeas.cs-net.gr/pub/unix/mcrypt/	ftp://argeas.cs-net.gr/pub/unix/mcrypt/
http://www.xe.net/iptc/	http://www.xe.net/iptc/
http://www.mysql.com/	http://www.mysql.com/
http://www.ifconnection.de/~tm/	http://www.ifconnection.de/~tm/
http://www.ifconnection.de/~tm/software/pdflib/PDFlib-0.6.pdf	http://www.ifconnection.de/~tm/software/pdflib/PDFlib-0.6.pdf
www.postgreSQL.org	http://www.postgresql.org/
RFC 1321	http://ds.internic.net/rfc/rfc1321.txt
http://www.qcc.sk.ca/~bguenter/distrib/vmailmgr/	http://www.qcc.sk.ca/~bguenter/distrib/vmailmgr/
WDDX	http://www.wddx.org/
http://www.cdrom.com/pub/infozip/zlib/	http://www.cdrom.com/pub/infozip/zlib/
http://www.w3.org/XML/	http://www.w3.org/XML/
http://www.jclark.com/xml/	http://www.jclark.com/xml/
http://www.guardian.no/~ssb/phpxml.html	http://www.guardian.no/~ssb/phpxml.html
Unicode	http://www.unicode.org/
section 4.2.2 of the XML 1.0 spec	http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent
section 4.7 of the XML 1.0 spec	