Drupal Reference Manual

4.5

Generated by Doxygen 1.3.5

Sun Apr 17 19:24:18 2005

Contents

Chapter 1

Drupal Module Index

1.1 Drupal Modules

Here is a list of all modules:

Chapter 2

Drupal File Index

2.1 Drupal File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Drupal Module Documentation

3.1 Input validation

Functions

- valid_email_address ($mail)
- valid_url ($url, $absolute=FALSE)
- valid_input_data ($data)

3.1.1 Detailed Description

Functions to validate user input.

3.1.2 Function Documentation

3.1.2.1 valid_email_address ($ *mail*)

Verify the syntax of the given e-mail address.

Empty e-mail addresses are allowed. See RFC 2822 for details.

Parameters:

$mail A string containing an email address.

Returns:

TRUE if the address is in a valid format.

3.1.2.2 valid_input_data ($ *data*)

Validate data input by a user.

Ensures that user data cannot be used to perform attacks on the site.

Parameters:

$data The input to check.

Returns:

TRUE if the input data is acceptable.

3.1.2.3 valid_url ($ *url*, $ *absolute* = FALSE)

Verify the syntax of the given URL.

Parameters:

$url The URL to verify.

$absolute Whether the URL is absolute (beginning with a scheme such as "http:").

Returns:

TRUE if the URL is in a valid format.

3.2 Search interface

Functions

- search_item ($item, $type)
- search_form ($action= ”, $keys= ”, $options=FALSE)
- search_data ($keys=NULL)
- search_type ($type, $action= ”, $keys= ”, $options=FALSE)

3.2.1 Detailed Description

The Drupal search interface manages a global search mechanism.

Modules may plug into this system to provide searches of different types of data. Most of the system is handled by search.module, so this must be enabled for all of the search features to work.

3.2.2 Function Documentation

3.2.2.1 search_data ($ *keys* = NULL)

Perform a global search on the given keys, and return the formatted results.

3.2.2.2 search_form ($ *action* = ”, $ *keys* = ”, $ *options* = FALSE)

Render a generic search form.

This form must be usable not only within "http://example.com/search", but also as a simple search box (without "Restrict search to", help text, etc.), in the theme’s header, and so forth. This means we must provide options to conditionally render certain parts of this form.

Parameters:

$action Form action. Defaults to "search".

$keys The search string entered by the user, containing keywords for the search.

$options Whether to render the optional form fields and text ("Restrict search to", help text, etc.).

Returns:

An HTML string containing the search form.

3.2.2.3 search_item ($ *item*, $ *type*)

Format a single result entry of a search query.

Modules may implement hook_search_item() in order to override this default function to display search results.

Parameters:

$item A single search result as returned by hook_search(). The result should be an array with keys "count", "link", "title", "user", "date", and "keywords".

$type The type of item found, such as "user" or "comment".

3.2.2.4 search_type ($ *type*, $ *action* = ”, $ *keys* = ”, $ *options* = FALSE)

Display a search form for a particular type of data.

Parameters:

$type The type of content to search within.

$action Form action. Defaults to "search".

$keys The search string entered by the user, containing keywords for the search.

$options Whether to render the optional form fields and text ("Restrict search to", help text, etc.).

Returns:

An HTML string containing the search form and results.

3.3 Formatting

Functions

- format_rss_channel ($title, $link, $description, $items, $language= 'en', $args=array())
- format_rss_item ($title, $link, $description, $args=array())
- format_plural ($count, $singular, $plural)
- format_size ($size)
- format_interval ($timestamp, $granularity=2)
- format_date ($timestamp, $type= 'medium', $format= '', $timezone=NULL)
- format_name ($object)

3.3.1 Detailed Description

Functions to format numbers, strings, dates, etc.

3.3.2 Function Documentation

3.3.2.1 format_date ($ *timestamp*, $ *type* = 'medium', $ *format* = '', $ *timezone* = NULL)

Format a date with the given configured format or a custom format string.

Drupal allows administrators to select formatting strings for 'small', 'medium' and 'large' date formats. This function can handle these formats, as well as any custom format.

Parameters:

$timestamp The exact date to format, as a UNIX timestamp.

$type The format to use. Can be "small", "medium" or "large" for the preconfigured date formats. If "custom" is specified, then $format is required as well.

$format A PHP date format string as required by date().

$timezone Time zone offset in seconds; if omitted, the user's time zone is used.

Returns:

A translated date string in the requested format.

3.3.2.2 format_interval ($ *timestamp*, $ *granularity* = 2)

Format a time interval with the requested granularity.

Parameters:

$timestamp The length of the interval in seconds.

$granularity How many different units to display in the string.

Returns:
A translated string representation of the interval.

3.3.2.3 format_name ($ *object*)

Format a username.

Parameters:
$object The user object to format, usually returned from user_load().

Returns:
A string containing an HTML link to the user's page if the passed object suggests that this is a site user. Otherwise, only the username is returned.

3.3.2.4 format_plural ($ *count*, $ *singular*, $ *plural*)

Format a string containing a count of items.

This function ensures that the string is pluralized correctly. Since t() is called by this function, make sure not to pass already-localized strings to it.

Parameters:
$count The item count to display.

$singular The string for the singular case. Please make sure it is clear this is singular, to ease translation (e.g. use "1 new comment" instead of "1 new").

$plural The string for the plural case. Please make sure it is clear this is plural, to ease translation. Use count in place of the item count, as in "%count new comments".

Returns:
A translated string.

3.3.2.5 format_rss_channel ($ *title*, $ *link*, $ *description*, $ *items*, $ *language* = 'en', $ *args* = array())

Formats an RSS channel.

Arbitrary elements may be added using the $args associative array.

3.3.2.6 format_rss_item ($ *title*, $ *link*, $ *description*, $ *args* = array())

Format a single RSS item.

Arbitrary elements may be added using the $args associative array.

3.3.2.7 format_size ($ *size*)

Generate a string representation for the given byte count.

Parameters:

$size The size in bytes.

Returns:

A translated string representation of the size.

3.4 Form generation

Functions

- form ($form, $method= 'post', $action=NULL, $attributes=NULL)
- form_set_error ($name, $message)
- form_get_errors ()
- _form_get_error ($name)
- **_form_get_class** ($name, $required, $error)
- form_item ($title, $value, $description=NULL, $id=NULL, $required=FALSE, $error=FALSE)
- form_group ($legend, $group, $description=NULL)
- form_radio ($title, $name, $value=1, $checked=FALSE, $description=NULL, $attributes=NULL, $required=FALSE)
- form_radios ($title, $name, $value, $options, $description=NULL, $required=FALSE, $attributes=NULL)
- form_checkbox ($title, $name, $value=1, $checked=FALSE, $description=NULL, $attributes=NULL, $required=FALSE)
- form_checkboxes ($title, $name, $values, $options, $description=NULL, $attributes=NULL, $required=FALSE)
- form_textfield ($title, $name, $value, $size, $maxlength, $description=NULL, $attributes=NULL, $required=FALSE)
- form_password ($title, $name, $value, $size, $maxlength, $description=NULL, $attributes=NULL, $required=FALSE)
- form_textarea ($title, $name, $value, $cols, $rows, $description=NULL, $attributes=NULL, $required=FALSE)
- form_select ($title, $name, $value, $options, $description=NULL, $extra=0, $multiple=FALSE, $required=FALSE)
- form_file ($title, $name, $size, $description=NULL, $required=FALSE)
- form_hidden ($name, $value)
- form_button ($value, $name= 'op', $type= 'submit', $attributes=NULL)
- form_submit ($value, $name= 'op', $attributes=NULL)
- form_weight ($title=NULL, $name= 'weight', $value=0, $delta=10, $description=NULL, $extra=0)

3.4.1 Detailed Description

Functions to enable output of HTML forms and form elements.

Drupal uses these functions to achieve consistency in its form presentation, while at the same time simplifying code and reducing the amount of HTML that must be explicitly generated by modules.

3.4.2 Function Documentation

3.4.2.1 _form_get_error ($ *name*)

Return the error message filed against the form with the specified name.

3.4.2.2 form ($ *form*, $ *method* = 'post', $ *action* = NULL, $ *attributes* = NULL)

Generate a form from a set of form elements.

Parameters:

$form An HTML string containing one or more form elements.

$method The query method to use ("post" or "get").

$action The URL to send the form contents to, if not the current page.

$attributes An associative array of attributes to add to the form tag.

Returns:

An HTML string with the contents of $form wrapped in a form tag.

3.4.2.3 form_button ($ *value*, $ *name* = 'op', $ *type* = 'submit', $ *attributes* = NULL)

Format an action button.

Parameters:

$value Both the label for the button, and the value passed to the target page when this button is clicked.

$name The internal name used to refer to the button.

$type What type to pass to the HTML input tag.

$attributes An associative array of HTML attributes to add to the form item.

Returns:

A themed HTML string representing the button.

3.4.2.4 form_checkbox ($ *title*, $ *name*, $ *value* = 1, $ *checked* = FALSE, $ *description* = NULL, $ *attributes* = NULL, $ *required* = FALSE)

Format a checkbox.

Parameters:

$title The label for the checkbox.

$name The internal name used to refer to the button.

$value The value that the form element takes on when selected.

$checked Whether the button will be initially selected when the page is rendered.

$description Explanatory text to display after the form item.

$attributes An associative array of HTML attributes to add to the button.

$required Whether the user must check this box before submitting the form.

Returns:

A themed HTML string representing the checkbox.

3.4.2.5 form_checkboxes ($ *title*, $ *name*, $ *values*, $ *options*, $ *description* = NULL, $ *attributes* = NULL, $ *required* = FALSE)

Format a set of checkboxes.

Parameters:

$title The label for the checkboxes as a group.

$name The internal name used to refer to the buttons.

$values A linear array of keys of the initially checked boxes.

$options An associative array of buttons to display. The keys in this array are button values, while the values are the labels to display for each button.

$description Explanatory text to display after the form item.

$attributes An associative array of HTML attributes to add to each button.

$required Whether the user must check a box before submitting the form.

Returns:

A themed HTML string representing the radio button set.

3.4.2.6 form_file ($ *title*, $ *name*, $ *size*, $ *description* = NULL, $ *required* = FALSE)

Format a file upload field.

Parameters:

$title The label for the file upload field.

$name The internal name used to refer to the field.

$size A measure of the visible size of the field (passed directly to HTML).

$description Explanatory text to display after the form item.

$required Whether the user must upload a file to the field.

Returns:

A themed HTML string representing the field.

For assistance with handling the uploaded file correctly, see the API provided by file.inc.

3.4.2.7 form_get_errors ()

Return an associative array of all errors.

3.4.2.8 form_group ($ *legend*, $ *group*, $ *description* = NULL)

Format a group of form items.

Parameters:

$legend The label for the form item group.

$group The form items within the group, as an HTML string.

$description Explanatory text to display after the form item group.

Returns:
A themed HTML string representing the form item group.

3.4.2.9 form_hidden ($ *name*, $ *value*)

Store data in a hidden form field.

Parameters:
$name The internal name used to refer to the field.

$value The stored data.

Returns:
A themed HTML string representing the hidden field.

This function can be useful in retaining information between page requests, but be sure to validate the data on the receiving page as it is possible for an attacker to change the value before it is submitted.

3.4.2.10 form_item ($ *title*, $ *value*, $ *description* = NULL, $ *id* = NULL, $ *required* = FALSE, $ *error* = FALSE)

Format a general form item.

Parameters:
$title The label for the form item.

$value The contents of the form item.

$description Explanatory text to display after the form item.

$id A unique identifier for the form item.

$required Whether the user must fill in this form element before submitting the form.

$error An error message to display alongside the form element.

Returns:
A themed HTML string representing the form item.

3.4.2.11 form_password ($ *title*, $ *name*, $ *value*, $ *size*, $ *maxlength*, $ *description* = NULL, $ *attributes* = NULL, $ *required* = FALSE)

Format a single-line text field that does not display its contents visibly.

Parameters:
$title The label for the text field.

$name The internal name used to refer to the field.

$value The initial value for the field at page load time.

$size A measure of the visible size of the field (passed directly to HTML).

$maxlength The maximum number of characters that may be entered in the field.

$description Explanatory text to display after the form item.

$attributes An associative array of HTML attributes to add to the form item.

$required Whether the user must enter some text in the field.

Returns:
A themed HTML string representing the field.

3.4.2.12 form_radio ($ *title*, $ *name*, $ *value* = 1, $ *checked* = FALSE, $ *description* = NULL, $ *attributes* = NULL, $ *required* = FALSE)

Format a radio button.

Parameters:

$title The label for the radio button.

$name The internal name used to refer to the button.

$value The value that the form element takes on when selected.

$checked Whether the button will be initially selected when the page is rendered.

$description Explanatory text to display after the form item.

$attributes An associative array of HTML attributes to add to the button.

$required Whether the user must select this radio button before submitting the form.

Returns:
A themed HTML string representing the radio button.

3.4.2.13 form_radios ($ *title*, $ *name*, $ *value*, $ *options*, $ *description* = NULL, $ *required* = FALSE, $ *attributes* = NULL)

Format a set of radio buttons.

Parameters:

$title The label for the radio buttons as a group.

$name The internal name used to refer to the buttons.

$value The currently selected radio button's key.

$options An associative array of buttons to display. The keys in this array are button values, while the values are the labels to display for each button.

$description Explanatory text to display after the form item.

$required Whether the user must select a radio button before submitting the form.

$attributes An associative array of HTML attributes to add to each button.

Returns:
A themed HTML string representing the radio button set.

3.4.2.14 form_select ($ *title*, $ *name*, $ *value*, $ *options*, $ *description* = NULL, $ *extra* = 0, $ *multiple* = FALSE, $ *required* = FALSE)

Format a dropdown menu or scrolling selection box.

Parameters:

$title The label for the form element.

$name The internal name used to refer to the form element.

$value The key of the currently selected item, or a linear array of keys of all the currently selected items if multiple selections are allowed.

$options An associative array of buttons to display. The keys in this array are button values, while the values are the labels to display for each button.

$description Explanatory text to display after the form item.

$extra Additional HTML to inject into the select element tag.

$multiple Whether the user may select more than one item.

$required Whether the user must select a value before submitting the form.

Returns:

A themed HTML string representing the form element.

It is possible to group options together; to do this, change the format of $options to an associative array in which the keys are group labels, and the values are associative arrays in the normal $options format.

3.4.2.15 form_set_error ($ *name*, $ *message*)

File an error against the form element with the specified name.

3.4.2.16 form_submit ($ *value*, $ *name* = 'op', $ *attributes* = NULL)

Format a form submit button.

Parameters:

$value Both the label for the button, and the value passed to the target page when this button is clicked.

$name The internal name used to refer to the button.

$attributes An associative array of HTML attributes to add to the form item.

Returns:

A themed HTML string representing the button.

3.4.2.17 form_textarea ($ *title*, $ *name*, $ *value*, $ *cols*, $ *rows*, $ *description* = NULL, $ *attributes* = NULL, $ *required* = FALSE)

Format a multiple-line text field.

Parameters:

$title The label for the text field.

$name The internal name used to refer to the field.

$value The initial value for the field at page load time.

$cols The width of the field, in columns of text.

$rows The height of the field, in rows of text.

$description Explanatory text to display after the form item.

$attributes An associative array of HTML attributes to add to the form item.

$required Whether the user must enter some text in the field.

Returns:

A themed HTML string representing the field.

3.4.2.18 form_textfield ($ *title*, $ *name*, $ *value*, $ *size*, $ *maxlength*, $ *description* = NULL, $ *attributes* = NULL, $ *required* = FALSE)

Format a single-line text field.

Parameters:

$title The label for the text field.

$name The internal name used to refer to the field.

$value The initial value for the field at page load time.

$size A measure of the visible size of the field (passed directly to HTML).

$maxlength The maximum number of characters that may be entered in the field.

$description Explanatory text to display after the form item.

$attributes An associative array of HTML attributes to add to the form item.

$required Whether the user must enter some text in the field.

Returns:

A themed HTML string representing the field.

3.4.2.19 form_weight ($ *title* = NULL, $ *name* = 'weight', $ *value* = 0, $ *delta* = 10, $ *description* = NULL, $ *extra* = 0)

Format a weight selection menu.

Parameters:

$title The label for the form element.

$name The internal name used to refer to the form element.

$value The selected weight value at page load time.

$delta The largest in absolute value the weight can be. For example, if set to 10, weights could range from -10 to 10 inclusive.

$description Explanatory text to display after the form item.

$extra Additional HTML to inject into the select element tag.

Returns:

A themed HTML string representing the form element.

3.5 Database abstraction layer

Functions

- db_prefix_tables ($sql)
- db_set_active ($name= 'default')
- db_connect ($url)
- pager_query ($query, $limit=10, $element=0, $count_query=NULL)
- tablesort_sql ($header, $before= '')

3.5.1 Detailed Description

Allow the use of different database servers using the same code base.

Drupal provides a slim database abstraction layer to provide developers with the ability to support multiple database servers easily. The intent of this layer is to preserve the syntax and power of SQL as much as possible, while letting Drupal control the pieces of queries that need to be written differently for different servers and provide basic security checks.

Most Drupal database queries are performed by a call to db_query() or db_query_range(). Module authors should also consider using pager_query() for queries that return results that need to be presented on multiple pages, and tablesort_sql() for generating appropriate queries for sortable tables.

For example, one might wish to return a list of the most recent 10 nodes authored by a given user. Instead of directly issuing the SQL query

```
SELECT n.title, n.body, n.created FROM node n WHERE n.uid = $uid LIMIT 0, 10;
```

one would instead call the Drupal functions:

```
$result = db_query_range('SELECT n.title, n.body, n.created
  FROM {node} n WHERE n.uid = %d', $uid, 0, 10);
while ($node = db_fetch_object($result)) {
  // Perform operations on $node->body, etc. here.
}
```

Curly braces are used around "node" to provide table prefixing via db_prefix_tables(). The explicit use of a user ID is pulled out into an argument passed to db_query() so that SQL injection attacks from user input can be caught and nullified. The LIMIT syntax varies between database servers, so that is abstracted into db_query_range() arguments. Finally, note the common pattern of iterating over the result set using db_fetch_object().

3.5.2 Function Documentation

3.5.2.1 db_connect ($ *url*)

Initialize a database connection.

Note that you can change the mysql_connect() call to mysql_pconnect() if you want to use persistent connections. This is not recommended on shared hosts, and might require additional database/webserver tuning. It can increase performance, however, when the overhead to connect to your database is high (e.g. your database and web server live on different machines).

3.5.2.2 db_prefix_tables ($ *sql*)

Append a database prefix to all tables in a query.

Queries sent to Drupal should wrap all table names in curly brackets. This function searches for this syntax and adds Drupal's table prefix to all tables, allowing Drupal to coexist with other systems in the same database if necessary.

Parameters:

$sql A string containing a partial or entire SQL query.

Returns:

The properly-prefixed string.

3.5.2.3 db_set_active ($ *name* = 'default')

Activate a database for future queries.

If it is necessary to use external databases in a project, this function can be used to change where database queries are sent. If the database has not yet been used, it is initialized using the URL specified for that name in Drupal's configuration file. If this name is not defined, a duplicate of the default connection is made instead.

Be sure to change the connection back to the default when done with custom code.

Parameters:

$name The name assigned to the newly active database connection. If omitted, the default connection will be made active.

3.5.2.4 pager_query ($ *query*, $ *limit* = 10, $ *element* = 0, $ *count_query* = NULL)

Perform a paged database query.

Use this function when doing select queries you wish to be able to page. The pager uses LIMIT-based queries to fetch only the records required to render a certain page. However, it has to learn the total number of records returned by the query to compute the number of pages (the number of records / records per page). This is done by inserting "COUNT(∗)" in the original query. For example, the query "SELECT nid, type FROM node WHERE status = '1' ORDER BY sticky DESC, created DESC" would be rewritten to read "SELECT COUNT(∗) FROM node WHERE status = '1' ORDER BY sticky DESC, created DESC". Rewriting the query is accomplished using a regular expression.

Unfortunately, the rewrite rule does not always work as intended for queries that already have a "COUNT(∗)" or a "GROUP BY" clause, and possibly for other complex queries. In those cases, you can optionally pass a query that will be used to count the records.

For example, if you want to page the query "SELECT COUNT(*), TYPE FROM node GROUP BY TYPE", pager_query() would invoke the incorrect query "SELECT COUNT(*) FROM node GROUP BY TYPE". So instead, you should pass "SELECT COUNT(DISTINCT(TYPE)) FROM node" as the optional $count_query parameter.

Parameters:

$query The SQL query that needs paging.

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$count_query An SQL query used to count matching records.

... A variable number of arguments which are substituted into the query (and also the count query) using printf() syntax.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

3.5.2.5 tablesort_sql ($ *header*, $ *before* = '')

Create an SQL sort clause.

This function produces the ORDER BY clause to insert in your SQL queries, assuring that the returned database table rows match the sort order chosen by the user.

Parameters:

$header An array of column headers in the format described in theme_table().

$before An SQL string to insert after ORDER BY and before the table sorting code. Useful for sorting by important attributes like "sticky" first.

Returns:

An SQL string to append to the end of a query.

3.6 File interface

Enumerations

- enum **FILE_DOWNLOADS_PUBLIC**
- enum **FILE_DOWNLOADS_PRIVATE**
- enum **FILE_CREATE_DIRECTORY**
- enum **FILE_MODIFY_PERMISSIONS**
- enum **FILE_DIRECTORY_TEMP**
- enum **FILE_EXISTS_RENAME**
- enum **FILE_EXISTS_REPLACE**
- enum **FILE_EXISTS_ERROR**

Functions

- file_create_url ($path)
- file_create_path ($dest=0)
- file_check_directory (&$directory, $mode=0, $form_item=NULL)
- file_check_path (&$path)
- file_check_upload ($source)
- file_check_location ($source, $directory=0)
- file_copy (&$source, $dest=0, $replace=FILE_EXISTS_RENAME)
- file_move (&$source, $dest=0, $replace=FILE_EXISTS_RENAME)
- **file_create_filename** ($basename, $directory)
- **file_delete** ($path)
- file_save_upload ($source, $dest=0, $replace=FILE_EXISTS_RENAME)
- file_save_data ($data, $dest, $replace=FILE_EXISTS_RENAME)
- file_transfer ($source, $headers)
- file_download ()
- file_scan_directory ($dir, $mask, $nomask=array('.', '..', 'CVS'), $callback=0, $recurse=TRUE)

3.6.1 Detailed Description

Common file handling functions.

3.6.2 Function Documentation

3.6.2.1 file_check_directory (&$ *directory*, $ *mode* = 0, $ *form_item* = NULL)

Check that directory exists and is writable.

Parameters:

$directory Path to extract and verify directory for.

$mode Try to create the directory if it does not exist.

$form_item Optional name for a field item to attach potential errors to.

Returns:
False when directory not found, or true when directory exists.

3.6.2.2 file_check_location ($ *source*, $ *directory* = 0)

Check if a file is really located inside $directory. Should be used to make sure a file specified is really located within the directory to prevent exploits.

```
// Returns false:
file_check_location('/www/example.com/files/../../../etc/passwd', '/www/example.com/files');
```

Parameters:
$source A string set to the file to check.

$directory A string where the file should be located.

Returns:
0 for invalid path or the real path of the source.

3.6.2.3 file_check_path (&$ *path*)

Checks path to see if it is a directory, or a dir/file.

Parameters:
$path

3.6.2.4 file_check_upload ($ *source*)

Check if $source is a valid file upload.

Parameters:
$source

3.6.2.5 file_copy (&$ *source*, $ *dest* = 0, $ *replace* = FILE_EXISTS_RENAME)

Copies a file to a new location. This is a powerful function that in many ways performs like an advanced version of copy().

- Checks if $source and $dest are valid and readable/writable.

- Performs a file copy if $source is not equal to $dest.
- If file already exists in $dest either the call will error out, replace the file or rename the file based on the $replace parameter.

Parameters:

$source A string specifying the file location of the original file. This parameter will contain the resulting destination filename in case of success.

$dest A string containing the directory $source should be copied to.

$replace Replace behavior when the destination file already exists.

- FILE_EXISTS_REPLACE - Replace the existing file
- FILE_EXISTS_RENAME - Append _{incrementing number} until the filename is unique
- FILE_EXISTS_ERROR - Do nothing and return false.

Returns:

True for success, false for failure.

3.6.2.6 file_create_path ($ *dest* = 0)

Make sure the destination is a complete path and resides in the file system directory, if it is not prepend the file system directory.

Parameters:

$dest Path to verify

Returns:

Path to file with file system directory appended if necessary.

3.6.2.7 file_create_url ($ *path*)

Create the download path to a file.

Parameters:

$path Path to the file to generate URL for

Returns:

URL pointing to the file

3.6.2.8 file_download ()

Call modules to find out if a file is accessible for a given user.

3.6.2.9 file_move (&$ *source*, $ *dest* = 0, $ *replace* = FILE_EXISTS_RENAME)

Moves a file to a new location.

- Checks if $source and $dest are valid and readable/writable.
- Performs a file move if $source is not equal to $dest.
- If file already exists in $dest either the call will error out, replace the file or rename the file based on the $replace parameter.

Parameters:

$source A string specifying the file location of the original file. This parameter will contain the resulting destination filename in case of success.

$dest A string containing the directory $source should be copied to.

$replace Replace behavior when the destination file already exists.

- FILE_EXISTS_REPLACE - Replace the existing file
- FILE_EXISTS_RENAME - Append _{incrementing number} until the filename is unique
- FILE_EXISTS_ERROR - Do nothing and return false.

Returns:

True for success, false for failure.

3.6.2.10 file_save_data ($ *data*, $ *dest*, $ *replace* = FILE_EXISTS_RENAME)

Save a string to the specified destination

Parameters:

$data A string containing the contents of the file

$dest A string containing the destination location

Returns:

A string containing the resulting filename or 0 on error

3.6.2.11 file_save_upload ($ *source*, $ *dest* = 0, $ *replace* = FILE_EXISTS_RENAME)

Saves a file upload to a new location. The source file is validated as a proper upload and handled as such.

Parameters:

$source A string specifying the name of the upload field to save. This parameter will contain the resulting destination filename in case of success.

$dest A string containing the directory $source should be copied to, will use the temporary directory in case no other value is set.

$replace A boolean, set to true if the destination should be replaced when in use, but when false append a _X to the filename.

Returns:

An object containing file info or 0 in case of error.

3.6.2.12 file_scan_directory ($ *dir*, $ *mask*, $ *nomask* = array('.', '..', 'CVS'), $ *callback* = 0, $ *recurse* = TRUE)

Finds all files that match a given mask in a given directory.

Parameters:

$dir Directory to scan

$mask Regular expression to filter out files. Only filenames that match the mask will be returned.

$nomask Array of filenames which should never be returned regardless if they match the $mask

$callback Function to call for qualifying file. Set to 0 or false if you do not want callbacks.

$recurse When true directory scan will recurse the entire tree starting at $dir

Returns:

Array of qualifying files

3.6.2.13 file_transfer ($ *source*, $ *headers*)

Transfer file using http to client. Pipes a file through Drupal to the client.

Parameters:

$source File to transfer.

$headers An array of http headers to send along with file.

3.7 Menu system

Menu flags

Flags for use in the "type" attribute of menu items.

- enum **MENU_IS_ROOT**
- enum **MENU_VISIBLE_IN_TREE**
- enum **MENU_VISIBLE_IN_BREADCRUMB**
- enum **MENU_VISIBLE_IF_HAS_CHILDREN**
- enum **MENU_MODIFIABLE_BY_ADMIN**
- enum **MENU_MODIFIED_BY_ADMIN**
- enum **MENU_CREATED_BY_ADMIN**
- enum **MENU_IS_LOCAL_TASK**
- enum **MENU_LINKS_TO_PARENT**

Menu item types

Menu item definitions provide one of these constants, which are shortcuts for combinations of the above flags.

- enum MENU_NORMAL_ITEM
- enum MENU_ITEM_GROUPING
- enum MENU_CALLBACK
- enum MENU_DYNAMIC_ITEM
- enum MENU_SUGGESTED_ITEM
- enum MENU_LOCAL_TASK
- enum MENU_DEFAULT_LOCAL_TASK
- enum MENU_CUSTOM_ITEM
- enum MENU_CUSTOM_MENU

Menu status codes

Status codes for menu callbacks.

- enum **MENU_FOUND**
- enum **MENU_NOT_FOUND**
- enum **MENU_ACCESS_DENIED**

Functions

- menu_get_menu ()
- menu_get_local_tasks ()
- menu_set_location ($location)
- menu_execute_active_handler ()

- menu_get_active_item ()
- menu_set_active_item ($path=NULL)
- menu_get_active_nontask_item ()
- menu_get_active_title ()
- menu_get_active_help ()
- menu_get_active_breadcrumb ()
- menu_in_active_trail ($mid)
- menu_rebuild ()

3.7.1 Detailed Description

Define the navigation menus, and route page requests to code based on URLs.

The Drupal menu system drives both the navigation system from a user perspective and the callback system that Drupal uses to respond to URLs passed from the browser. For this reason, a good understanding of the menu system is fundamental to the creation of complex modules.

Drupal's menu system follows a simple hierarchy defined by paths. Implementations of hook_menu() define menu items and assign them to paths (which should be unique). The menu system aggregates these items and determines the menu hierarchy from the paths. For example, if the paths defined were a, a/b, e, a/b/c/d, f/g, and a/b/h, the menu system would form the structure:

- a
  - a/b
    - a/b/c/d
    - a/b/h
- e
- f/g Note that the number of elements in the path does not necessarily determine the depth of the menu item in the tree.

When responding to a page request, the menu system looks to see if the path requested by the browser is registered as a menu item with a callback. If not, the system searches up the menu tree for the most complete match with a callback it can find. If the path a/b/i is requested in the tree above, the callback for a/b would be used.

The found callback function is called with any arguments specified in the "callback arguments" attribute of its menu item. After these arguments, any remaining components of the path are appended as further arguments. In this way, the callback for a/b above could respond to a request for a/b/i differently than a request for a/b/j.

For an illustration of this process, see page_example.module.

Access to the callback functions is also protected by the menu system. The "access" attribute of each menu item is checked as the search for a callback proceeds. If this attribute is TRUE, then access is granted; if FALSE, then access is denied. The first found "access" attribute determines the accessibility of the target. Menu items may omit this attribute to use the value provided by an ancestor item.

In the default Drupal interface, you will notice many links rendered as tabs. These are known in the menu system as "local tasks", and they are rendered as tabs by default, though other presentations are possible. Local tasks function just as other menu items in most respects. It is convention that the names of these tasks should be short verbs if possible. In addition, a "default" local task should be provided for each set. When visiting a local task's parent menu item, the default local task will be rendered as if it is selected; this provides for a normal tab user experience. This default task is special in that it links not to its provided path, but to its parent item's path instead. The default task's path is only used to place it appropriately in the menu hierarchy.

3.7.2 Enumeration Type Documentation

3.7.2.1 enum MENU_CALLBACK

Callbacks simply register a path so that the correct function is fired when the URL is accessed. They are not shown in the menu.

3.7.2.2 enum MENU_CUSTOM_ITEM

Custom items are those defined by the administrator. Reserved for internal use; do not return from hook_-menu() implementations.

3.7.2.3 enum MENU_CUSTOM_MENU

Custom menus are those defined by the administrator. Reserved for internal use; do not return from hook_-menu() implementations.

3.7.2.4 enum MENU_DEFAULT_LOCAL_TASK

Every set of local tasks should provide one "default" task, that links to the same path as its parent when clicked.

3.7.2.5 enum MENU_DYNAMIC_ITEM

Dynamic menu items change frequently, and so should not be stored in the database for administrative customization.

3.7.2.6 enum MENU_ITEM_GROUPING

Item groupings are used for pages like "node/add" that simply list subpages to visit. They are distinguished from other pages in that they will disappear from the menu if no subpages exist.

3.7.2.7 enum MENU_LOCAL_TASK

Local tasks are rendered as tabs by default. Use this for menu items that describe actions to be performed on their parent item. An example is the path "node/52/edit", which performs the "edit" task on "node/52".

3.7.2.8 enum MENU_NORMAL_ITEM

Normal menu items show up in the menu tree and can be moved/hidden by the administrator. Use this for most menu items. It is the default value if no menu item type is specified.

3.7.2.9 enum MENU_SUGGESTED_ITEM

Modules may "suggest" menu items that the administrator may enable. They act just as callbacks do until enabled, at which time they act like normal items.

3.7.3 Function Documentation

3.7.3.1 menu_execute_active_handler ()

Execute the handler associated with the active menu item.

This is called early in the page request. The active menu item is at this point determined exclusively by the URL. The handler that is called here may, as a side effect, change the active menu item so that later menu functions (that display the menus and breadcrumbs, for example) act as if the user were in a different location on the site.

3.7.3.2 menu_get_active_breadcrumb ()

Returns an array of rendered menu items in the active breadcrumb trail.

3.7.3.3 menu_get_active_help ()

Returns the help associated with the active menu item.

3.7.3.4 menu_get_active_item ()

Returns the ID of the active menu item.

3.7.3.5 menu_get_active_nontask_item ()

Returns the ID of the current menu item or, if the current item is a local task, the menu item to which this task is attached.

3.7.3.6 menu_get_active_title ()

Returns the title of the active menu item.

3.7.3.7 menu_get_local_tasks ()

Return the local task tree.

Unlike the rest of the menu structure, the local task tree cannot be cached nor determined too early in the page request, because the user's current location may be changed by a menu_set_location() call, and the tasks shown (just as the breadcrumb trail) need to reflect the changed location.

3.7.3.8 menu_get_menu ()

Return the menu data structure.

The returned structure contains much information that is useful only internally in the menu system. External modules are likely to need only the ['visible'] element of the returned array. All menu items that

are accessible to the current user and not hidden will be present here, so modules and themes can use this structure to build their own representations of the menu.

$menu['visible'] will contain an associative array, the keys of which are menu IDs. The values of this array are themselves associative arrays, with the following key-value pairs defined:

- 'title' - The displayed title of the menu or menu item. It will already have been translated by the locale system.
- 'description' - The description (link title attribute) of the menu item. It will already have been translated by the locale system.
- 'path' - The Drupal path to the menu item. A link to a particular item can thus be constructed with l($item['title'], $item['path'], array('title' => $item['description'])).
- 'children' - A linear list of the menu ID's of this item's children.

Menu ID 0 is the "root" of the menu. The children of this item are the menus themselves (they will have no associated path). Menu ID 1 will always be one of these children; it is the default "Navigation" menu.

3.7.3.9 menu_in_active_trail ($ *mid*)

Returns true when the menu item is in the active trail.

3.7.3.10 menu_rebuild ()

Populate the database representation of the menu.

This need only be called at the start of pages that modify the menu.

3.7.3.11 menu_set_active_item ($ *path* = NULL)

Sets the path of the active menu item.

3.7.3.12 menu_set_location ($ *location*)

Change the current menu location of the user.

Frequently, modules may want to make a page or node act as if it were in the menu tree somewhere, even though it was not registered in a hook_menu() implementation. If the administrator has rearranged the menu, the newly set location should respect this in the breadcrumb trail and expanded/collapsed status of menu items in the tree. This function allows this behavior.

Parameters:

$location An array specifying a complete or partial breadcrumb trail for the new location, in the same format as the return value of hook_menu(). The last element of this array should be the new location itself.

This function will set the new breadcrumb trail to the passed-in value, but if any elements of this trail are visible in the site tree, the trail will be "spliced in" to the existing site navigation at that point.

3.8 Hooks

Functions

- module_hook ($module, $hook)
- module_invoke ($module, $hook, $a1=NULL, $a2=NULL, $a3=NULL, $a4=NULL)
- module_invoke_all ($hook, $a1=NULL, $a2=NULL, $a3=NULL, $a4=NULL)

3.8.1 Detailed Description

Allow modules to interact with the Drupal core.

Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named foo_-bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook. Each hook has a defined set of parameters and a specified result type.

To extend Drupal, a module need simply implement a hook. When Drupal wishes to allow intervention from modules, it determines which modules implement a hook and call that hook in all enabled modules that implement it.

The available hooks to implement are explained here in the Hooks section of the developer documentation. The string "hook" is used as a placeholder for the module name is the hook definitions. For example, if the module file is called example.module, then hook_help() as implemented by that module would be defined as example_help().

3.8.2 Function Documentation

3.8.2.1 module_hook ($ *module*, $ *hook*)

Determine whether a module implements a hook.

Parameters:

$module The name of the module (without the .module extension).

$hook The name of the hook (e.g. "help" or "menu").

Returns:

TRUE if the module is both installed and enabled, and the hook is implemented in that module.

3.8.2.2 module_invoke ($ *module*, $ *hook*, $ *a1* = NULL, $ *a2* = NULL, $ *a3* = NULL, $ *a4* = NULL)

Invoke a hook in a particular module.

Parameters:

$module The name of the module (without the .module extension).

$hook The name of the hook to invoke.

... Arguments to pass to the hook implementation.

Returns:
The return value of the hook implementation.

3.8.2.3 module_invoke_all ($ *hook*, $ *a1* = NULL, $ *a2* = NULL, $ *a3* = NULL, $ *a4* = NULL)

Invoke a hook in all enabled modules that implement it.

Parameters:
$hook The name of the hook to invoke.

... Arguments to pass to the hook.

Returns:
An array of return values of the hook implementations. If modules return arrays from their implementations, those are merged into one array.

3.9 Themeable functions

Functions

- theme_menu_tree ($pid=1, $all=FALSE)
- theme_menu_item ($mid)
- theme_menu_local_tasks ()
- theme_menu_local_task ($mid, $active)
- theme_pager ($tags=array(), $limit=10, $element=0, $attributes=array())
- theme_page ($content, $title=NULL, $breadcrumb=NULL)
- theme_status_messages ()
- theme_links ($links, $delimiter= '| ')
- theme_image ($path, $alt= '', $title= '', $attr= '', $getsize=true)
- theme_breadcrumb ($breadcrumb)
- theme_node ($node, $teaser=FALSE, $page=FALSE)
- theme_form_element ($title, $value, $description=NULL, $id=NULL, $required=FALSE, $error=FALSE)
- theme_submenu ($links)
- theme_table ($header, $rows, $attributes=NULL)
- theme_box ($title, $content, $region= 'main')
- theme_block ($block)
- theme_mark ()
- theme_stylesheet_import ($stylesheet, $media= 'all')
- theme_item_list ($items=array(), $title=NULL)
- theme_error ($message)
- **theme_more_help_link** ($url)
- theme_xml_icon ($url)
- theme_closure ($main=0)
- theme_onload_attribute ($theme_onloads=array())
- theme_blocks ($region)
- function theme_aggregator_feed ($feed)
- function theme_aggregator_block_item ($item, $feed=0)
- function theme_aggregator_summary_item ($item)
- function theme_aggregator_page_item ($item)
- function theme_filter_tips ($tips, $long=false, $extra= '')
- function theme_forum_display ($forums, $topics, $parents, $tid, $sortby, $forum_per_page)
- function theme_forum_list ($forums, $parents, $tid)
- function theme_forum_topic_list ($tid, $topics, $sortby, $forum_per_page)

3.9.1 Detailed Description

Functions that display HTML, and which can be customized by themes.

All functions that produce HTML for display should be themeable. This means that they should be named with the theme_ prefix, and invoked using theme() rather than being called directly. This allows themes to override the display of any Drupal object.

The theme system is described and defined in theme.inc.

3.9.2 Function Documentation

3.9.2.1 function theme_aggregator_block_item ($ *item*, $ *feed* = 0)

Format an individual feed item for display in the block.

3.9.2.2 function theme_aggregator_feed ($ *feed*)

Format a news feed.

3.9.2.3 function theme_aggregator_page_item ($ *item*)

Format an individual feed item for display on the aggregator page.

3.9.2.4 function theme_aggregator_summary_item ($ *item*)

Return a themed item heading for summary pages located at "aggregator/sources" and "aggregator/categories".

Parameters:

$item The item object from the aggregator module.

Returns:

A string containing the output.

3.9.2.5 theme_block ($ *block*)

Return a themed block.

You can style your blocks by defining .block (all blocks), .block-*module* (all blocks of module *module*), and #block-*module*-*delta* (specific block of module *module* with delta *delta*) in your theme's CSS.

Parameters:

$block An object populated with fields from the "blocks" database table ($block->module, $block->delta, $block->region, ...) and fields returned by *module*_block('view') ($block->subject, $block->content, ...).

Returns:

A string containing the block output.

3.9.2.6 theme_blocks ($ *region*)

Return a set of blocks available for the current user.

Parameters:
$region Which set of blocks to retrieve.

Returns:
A string containing the themed blocks for this region.

3.9.2.7 theme_box ($ *title*, $ *content*, $ *region* = 'main')

Return a themed box.

Parameters:
$title The subject of the box.

$content The content of the box.

$region The region in which the box is displayed.

Returns:
A string containing the box output.

3.9.2.8 theme_breadcrumb ($ *breadcrumb*)

Return a themed breadcrumb trail.

Parameters:
$breadcrumb An array containing the breadcrumb links.

Returns:
a string containing the breadcrumb output.

3.9.2.9 theme_closure ($ *main* = 0)

Execute hook_footer() which is run at the end of the page right before the close of the body tag.

Parameters:
$main (optional)

Returns:
A string containing the results of the hook_footer() calls.

3.9.2.10 theme_error ($ *message*)

Return a themed error message. REMOVE: this function is deprecated an no longer used in core.

Parameters:

$message The error message to be themed.

Returns:

A string containing the error output.

3.9.2.11 function theme_filter_tips ($ *tips*, $ *long* = false, $ *extra* = '')

Format a set of filter tips.

3.9.2.12 theme_form_element ($ *title*, $ *value*, $ *description* = NULL, $ *id* = NULL, $ *required* = FALSE, $ *error* = FALSE)

Return a themed form element.

Parameters:

$title the form element's title

$value the form element's data

$description the form element's description or explanation

$id the form element's ID used by the <label> tag

$required a boolean to indicate whether this is a required field or not

$error a string with an error message filed against this form element

Returns:

a string representing the form element

3.9.2.13 function theme_forum_display ($ *forums*, $ *topics*, $ *parents*, $ *tid*, $ *sortby*, $ *forum_per_page*)

Format the forum body.

3.9.2.14 function theme_forum_list ($ *forums*, $ *parents*, $ *tid*)

Format the forum listing.

3.9.2.15 function theme_forum_topic_list ($ *tid*, $ *topics*, $ *sortby*, $ *forum_per_page*)

Format the topic listing.

3.9.2.16 theme_image ($ *path*, $ *alt* = '', $ *title* = '', $ *attr* = '', $ *getsize* = true)

Return a themed image.

Parameters:

$path The path of the image file.

$alt The alternative text for text-based browsers.

$title The title text is displayed when the image is hovered in some popular browsers.

$attr Attributes placed in the img tag.

$getsize If set to true, the image's dimension are fetched and added as width/height attributes.

Returns:

A string containing the image tag.

3.9.2.17 theme_item_list ($ *items* = array(), $ *title* = NULL)

Return a themed list of items.

Parameters:

$items An array of items to be displayed in the list.

$title The title of the list.

Returns:

A string containing the list output.

3.9.2.18 theme_links ($ *links*, $ *delimiter* = ' | ')

Return a themed set of links.

Parameters:

$links An array of links to be themed.

$delimiter A string used to separate the links.

Returns:

A string containing the themed links.

3.9.2.19 theme_mark ()

Return a themed marker, useful for marking new comments or required form elements.

Returns:

A string containing the marker.

3.9.2.20 theme_menu_item ($ *mid*)

Generate the HTML representing a given menu item ID.

Parameters:

$mid The menu ID to render.

3.9.2.21 theme_menu_local_task ($ *mid*, $ *active*)

Generate the HTML representing a given menu item ID as a tab.

Parameters:

$mid The menu ID to render.

$active Whether this tab or a subtab is the active menu item.

3.9.2.22 theme_menu_local_tasks ()

Returns the rendered local tasks. The default implementation renders them as tabs.

3.9.2.23 theme_menu_tree ($ *pid* = 1, $ *all* = FALSE)

Returns a rendered menu tree.

3.9.2.24 theme_node ($ *node*, $ *teaser* = FALSE, $ *page* = FALSE)

Return a themed node.

Parameters:

$node An object providing all relevant information for displaying a node:

- $node->nid: The ID of the node.
- $node->type: The content type (story, blog, forum...).
- $node->title: The title of the node.
- $node->created: The creation date, as a UNIX timestamp.
- $node->teaser: A shortened version of the node body.
- $node->body: The entire node contents.
- $node->changed: The last modification date, as a UNIX timestamp.
- $node->uid: The ID of the author.
- $node->username: The username of the author.

$teaser Whether to display the teaser only, as on the main page.

$page Whether to display the node as a standalone page. If TRUE, do not display the title because it will be provided by the menu system.

Returns:

A string containing the node output.

3.9.2.25 theme_onload_attribute ($ *theme_onloads* = array())

Call hook_onload() in all modules to enable modules to insert JavaScript that will get run once the page has been loaded by the browser.

Parameters:

$theme_onloads Additional onload directives.

Returns:

A string containing the onload attributes.

3.9.2.26 theme_page ($ *content*, $ *title* = NULL, $ *breadcrumb* = NULL)

Return an entire Drupal page displaying the supplied content.

Parameters:

$content A string to display in the main content area of the page.

$title The title of the page, if different from that provided by the menu system.

$breadcrumb The breadcrumb trail for the page, if different from that provided by the menu system. Use menu_set_location() instead, if possible.

Returns:

A string containing the entire HTML page.

3.9.2.27 theme_pager ($ *tags* = array(), $ *limit* = 10, $ *element* = 0, $ *attributes* = array())

Format a query pager.

Menu callbacks that display paged query results should call theme('pager') to retrieve a pager control so that users can view other results.

Parameters:

$tags An array of labels for the controls in the pager.

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$attributes An associative array of query string parameters to append to the pager links.

Returns:

An HTML string that generates the query pager.

3.9.2.28 theme_status_messages ()

Returns themed set of status and/or error messages. The messages are grouped by type.

Returns:

A string containing the messages.

3.9.2.29 theme_stylesheet_import ($ *stylesheet*, $ *media* = 'all')

Import a stylesheet using .

Parameters:

$stylesheet The filename to point the link at.

$media The media type to specify for the stylesheet

Returns:

A string containing the HTML for the stylesheet import.

3.9.2.30 theme_submenu ($ *links*)

Return a themed submenu, typically displayed under the tabs.

Parameters:

$links An array of links.

3.9.2.31 theme_table ($ *header*, $ *rows*, $ *attributes* = NULL)

Return a themed table.

Parameters:

$header An array containing the table headers. Each element of the array can be either a localized string or an associative array with the following keys:

- "data": The localized title of the table column.
- "field": The database field represented in the table column (required if user is to be able to sort on this column).
- "sort": A default sort order for this column ("asc" or "desc").
- Any HTML attributes, such as "colspan", to apply to the column header cell.

$rows An array of table rows. Every row is an array of cells, or an associative array with the following keys:

- "data": an array of cells
- Any HTML attributes, such as "class", to apply to the table row.

Each cell can be either a string or an associative array with the following keys:

- "data": The string to display in the table cell.
- Any HTML attributes, such as "colspan", to apply to the table cell.

Here's an example for $rows:

```
*   $rows = array(
*     // Simple row
*     array(
*       'Cell 1', 'Cell 2', 'Cell 3'
*     ),
*     // Row with attributes on the row and some of its cells.
*     array(
*       'data' => array('Cell 1', array('data' => 'Cell 2', 'colspan' => 2)), 'class' => 'funky'
*     )
*   );
*
```

Parameters:

$attributes An array of HTML attributes to apply to the table tag.

Returns:

An HTML string representing the table.

3.9.2.32 theme_xml_icon ($ *url*)

Return code that emits an XML icon.

3.10 Node access rights

Functions

- function node_access ($op, $node=NULL, $uid=NULL)
- function node_access_join_sql ($node_alias= 'n', $node_access_alias= 'na')
- function node_access_where_sql ($op= 'view', $node_access_alias= 'na', $uid=NULL)
- function node_access_grants ($op, $uid=NULL)

3.10.1 Detailed Description

The node access system determines who can do what to which nodes.

In determining access rights for a node, node_access() first checks whether the user has the "administer nodes" permission. Such users have unrestricted access to all nodes. Then the node module's hook_access() is called, and a TRUE or FALSE return value will grant or deny access. This allows, for example, the blog module to always grant access to the blog author, and for the book module to always deny editing access to PHP pages.

If node module does not intervene (returns NULL), then the node_access table is used to determine access. All node access modules are queried using hook_node_grants() to assemble a list of "grant IDs" for the user. This list is compared against the table. If any row contains the node ID in question (or 0, which stands for "all nodes"), one of the grant IDs returned, and a value of TRUE for the operation in question, then access is granted. Note that this table is a list of grants; any matching row is sufficient to grant access to the node.

In node listings, the process above is followed except that hook_access() is not called on each node for performance reasons and for proper functioning of the pager system. When adding a node listing to your module, be sure to use node_access_join_sql() and node_access_where_sql() to add the appropriate clauses to your query for access checks.

To see how to write a node access module of your own, see node_access_example.module.

3.10.2 Function Documentation

3.10.2.1 function node_access ($ *op*, $ *node* = NULL, $ *uid* = NULL)

Determine whether the current user may perform the given operation on the specified node.

Parameters:

$op The operation to be performed on the node. Possible values are:

- "view"
- "update"
- "delete"

$node The node object (or node array) on which the operation is to be performed.

$uid The user ID on which the operation is to be performed.

Returns:
TRUE if the operation may be performed.

3.10.2.2 function node_access_grants ($ *op*, $ *uid* = NULL)

Fetch an array of permission IDs granted to the given user ID.

The implementation here provides only the universal "all" grant. A node access module should implement hook_node_grants() to provide a grant list for the user.

Parameters:
$op The operation that the user is trying to perform.

$uid The user ID performing the operation. If omitted, the current user is used.

Returns:
An associative array in which the keys are realms, and the values are arrays of grants for those realms.

3.10.2.3 function node_access_join_sql ($ *node_alias* = 'n', $ *node_access_alias* = 'na')

Generate an SQL join clause for use in fetching a node listing.

Parameters:
$node_alias If the node table has been given an SQL alias other than the default "n", that must be passed here.

$node_access_alias If the node_access table has been given an SQL alias other than the default "na", that must be passed here.

Returns:
An SQL join clause.

3.10.2.4 function node_access_where_sql ($ *op* = 'view', $ *node_access_alias* = 'na', $ *uid* = NULL)

Generate an SQL where clause for use in fetching a node listing.

Parameters:
$op The operation that must be allowed to return a node.

$node_access_alias If the node_access table has been given an SQL alias other than the default "na", that must be passed here.

Returns:
An SQL where clause.

Chapter 4

Drupal File Documentation

4.1 bootstrap.inc File Reference

Enumerations

- enum **CACHE_PERMANENT**
- enum **CACHE_TEMPORARY**

Functions

- conf_init ()
- variable_init ($conf=array())
- variable_get ($name, $default)
- variable_set ($name, $value)
- variable_del ($name)
- cache_get ($key)
- cache_set ($cid, $data, $expire=CACHE_PERMANENT, $headers=NULL)
- cache_clear_all ($cid=NULL, $wildcard=false)
- page_set_cache ()
- page_get_cache ()
- drupal_page_header ()
- bootstrap_invoke_all ($op)
- bootstrap_hooks ()
- drupal_get_path_alias ($path)
- drupal_get_path_map ($action= ”)
- drupal_set_title ($title=NULL)
- drupal_get_title ()
- drupal_unpack ($obj, $field= ’data’)
- referer_uri ()
- arg ($index)
- check_query ($text)
- check_url ($uri)

- request_uri ()
- timer_start ()
- watchdog ($type, $message, $link=NULL)
- drupal_set_message ($message=NULL, $type= 'status')
- drupal_get_messages ()

Variables

- **$config** = conf_init()
- **$conf** = variable_init(isset($conf) ? $conf : array())

4.1.1 Detailed Description

Functions that need to be loaded on every Drupal request.

4.1.2 Function Documentation

4.1.2.1 arg ($ *index*)

Return a component of the current Drupal path.

When viewing a page at the path "admin/node/configure", for example, arg(0) would return "admin", arg(1) would return "node", and arg(2) would return "configure".

Avoid use of this function where possible, as resulting code is hard to read. Instead, attempt to use named arguments in menu callback functions. See the explanation in menu.inc for how to construct callbacks that take arguments.

4.1.2.2 bootstrap_hooks ()

Define the critical hooks that force modules to always be loaded.

4.1.2.3 bootstrap_invoke_all ($ *op*)

Call all init or exit hooks without including all modules.

Parameters:

$op The name of the bootstrap hook we wish to invoke.

4.1.2.4 cache_clear_all ($ *cid* = NULL, $ *wildcard* = false)

Expire data from the cache.

Parameters:

$cid If set, the cache ID to delete. Otherwise, all cache entries that can expire are deleted.

$wildcard If set to true, the $cid is treated as a substring to match rather than a complete ID.

4.1.2.5 cache_get ($ *key*)

Return data from the persistent cache.

Parameters:

$key The cache ID of the data to retrieve.

4.1.2.6 cache_set ($ *cid*, $ *data*, $ *expire* = CACHE_PERMANENT, $ *headers* = NULL)

Store data in the persistent cache.

Parameters:

$cid The cache ID of the data to store.

$data The data to store in the cache. Complex data types must be serialized first.

$expire One of the following values:

- CACHE_PERMANENT: Indicates that the item should never be removed unless explicitly told to using cache_clear_all() with a cache ID.
- CACHE_TEMPORARY: Indicates that the item should be removed at the next general cache wipe.
- A Unix timestamp: Indicates that the item should be kept at least until the given time, after which it behaves like CACHE_TEMPORARY.

$headers A string containing HTTP header information for cached pages.

4.1.2.7 check_query ($ *text*)

Prepare user input for use in a database query, preventing SQL injection attacks.

4.1.2.8 check_url ($ *uri*)

Prepare user input for use in a URI.

We replace (and) with their entity equivalents to prevent XSS attacks.

4.1.2.9 conf_init ()

Locate the appropriate configuration file.

Try finding a matching configuration file by stripping the website's URI from left to right. If no configuration file is found, return the default value, "conf".

4.1.2.10 drupal_get_messages ()

Return all messages that have been set.

As a side effect, this function clears the message queue.

4.1.2.11 drupal_get_path_alias ($ *path*)

Given an internal Drupal path, return the alias set by the administrator.

4.1.2.12 drupal_get_path_map ($ *action* = ")

Return an array mapping path aliases to their internal Drupal paths.

4.1.2.13 drupal_get_title ()

Get the title of the current page, for display on the page and in the title bar.

4.1.2.14 drupal_page_header ()

Set HTTP headers in preparation for a page response.

4.1.2.15 drupal_set_message ($ *message* = NULL, $ *type* = 'status')

Set a message for the user to see.

The message is stored in the session so that it can persist through a redirect.

If called with no arguments, this function returns all set messages without clearing them.

4.1.2.16 drupal_set_title ($ *title* = NULL)

Set the title of the current page, for display on the page and in the title bar.

4.1.2.17 drupal_unpack ($ *obj*, $ *field* = 'data')

Unserializes and appends elements from a serialized string.

Parameters:

$obj The object to which the elements are appended.

$field The attribute of $obj whose value should be unserialized.

4.1.2.18 page_get_cache ()

Retrieve the current page from the cache.

Note, we do not serve cached pages when status messages are waiting (from a redirected form submission which was completed). Because the output handler is not activated, the resulting page will not get cached either.

4.1.2.19 page_set_cache ()

Store the current page in the cache.

4.1.2.20 referer_uri ()

Return the URI of the referring page.

4.1.2.21 request_uri ()

Since request_uri() is only available on Apache, we generate an equivalent using other environment vars.

4.1.2.22 timer_start ()

Begin a global timer, for benchmarking of page execution time.

4.1.2.23 variable_del ($ *name*)

Unset a persistent variable.

Parameters:

$name The name of the variable to undefine.

4.1.2.24 variable_get ($ *name*, $ *default*)

Return a persistent variable.

Parameters:

$name The name of the variable to return.

$default The default value to use if this variable has never been set.

Returns:

The value of the variable.

4.1.2.25 variable_init ($ *conf* = array())

Load the persistent variable table.

The variable table is composed of values that have been saved in the table with variable_set() as well as those explicitly specified in the configuration file.

4.1.2.26 variable_set ($ *name*, $ *value*)

Set a persistent variable.

Parameters:

$name The name of the variable to set.

$value The value to set. This can be any PHP data type; these functions take care of serialization as necessary.

4.1.2.27 watchdog ($ *type*, $ *message*, $ *link* = NULL)

Log a system message.

Parameters:

$type The category to which this message belongs.

$message The message to store in the log.

$link A link to associate with the message.

4.2 common.inc File Reference

HTTP handling

Functions to properly handle HTTP responses.

- drupal_goto ($path= ”, $query=NULL, $fragment=NULL)
- drupal_not_found ()
- drupal_access_denied ()
- drupal_http_request ($url, $headers=array(), $method= ’GET’, $data=NULL, $retry=3)

Conversion

Converts data structures to different types.

- array2object ($array)
- object2array ($object)

Messages

Frequently used messages.

- message_access ()
- message_na ()

Functions

- drupal_set_breadcrumb ($breadcrumb=NULL)
- drupal_get_breadcrumb ()
- drupal_set_html_head ($data=NULL)
- drupal_get_html_head ()
- drupal_rebuild_path_map ()
- drupal_get_normal_path ($path)
- drupal_set_header ($header=NULL)
- drupal_get_headers ()
- error_handler ($errno, $message, $filename, $line, $variables)
- **_fix_gpc_magic** (&$item, $key)
- fix_gpc_magic ()
- locale_initialize ()
- t ($string, $args=0)
- drupal_specialchars ($input, $quotes=ENT_NOQUOTES)
- valid_email_address ($mail)
- valid_url ($url, $absolute=FALSE)
- valid_input_data ($data)
- search_item ($item, $type)

- search_form ($action= ”, $keys= ”, $options=FALSE)
- search_data ($keys=NULL)
- search_type ($type, $action= ”, $keys= ”, $options=FALSE)
- check_form ($text)
- **check_file** ($filename)
- format_rss_channel ($title, $link, $description, $items, $language= ’en’, $args=array())
- format_rss_item ($title, $link, $description, $args=array())
- format_plural ($count, $singular, $plural)
- format_size ($size)
- format_interval ($timestamp, $granularity=2)
- format_date ($timestamp, $type= ’medium’, $format= ”, $timezone=NULL)
- format_name ($object)
- form ($form, $method= ’post’, $action=NULL, $attributes=NULL)
- form_set_error ($name, $message)
- form_get_errors ()
- _form_get_error ($name)
- **_form_get_class** ($name, $required, $error)
- form_item ($title, $value, $description=NULL, $id=NULL, $required=FALSE, $error=FALSE)
- form_group ($legend, $group, $description=NULL)
- form_radio ($title, $name, $value=1, $checked=FALSE, $description=NULL, $attributes=NULL, $required=FALSE)
- form_radios ($title, $name, $value, $options, $description=NULL, $required=FALSE, $attributes=NULL)
- form_checkbox ($title, $name, $value=1, $checked=FALSE, $description=NULL, $attributes=NULL, $required=FALSE)
- form_checkboxes ($title, $name, $values, $options, $description=NULL, $attributes=NULL, $required=FALSE)
- form_textfield ($title, $name, $value, $size, $maxlength, $description=NULL, $attributes=NULL, $required=FALSE)
- form_password ($title, $name, $value, $size, $maxlength, $description=NULL, $attributes=NULL, $required=FALSE)
- form_textarea ($title, $name, $value, $cols, $rows, $description=NULL, $attributes=NULL, $required=FALSE)
- form_select ($title, $name, $value, $options, $description=NULL, $extra=0, $multiple=FALSE, $required=FALSE)
- form_file ($title, $name, $size, $description=NULL, $required=FALSE)
- form_hidden ($name, $value)
- form_button ($value, $name= ’op’, $type= ’submit’, $attributes=NULL)
- form_submit ($value, $name= ’op’, $attributes=NULL)
- form_weight ($title=NULL, $name= ’weight’, $value=0, $delta=10, $description=NULL, $extra=0)
- url ($path=NULL, $query=NULL, $fragment=NULL, $absolute=FALSE)
- drupal_attributes ($attributes=array())
- l ($text, $path, $attributes=array(), $query=NULL, $fragment=NULL, $absolute=FALSE)
- **field_get** ($string, $name)
- **field_set** ($string, $name, $value)
- link_page ()
- link_node ($node, $main=0)
- drupal_page_footer ()
- drupal_map_assoc ($array, $function=NULL)
- drupal_xml_parser_create (&$data)

4.2.1 Detailed Description

Common functions that many Drupal modules will need to reference.

The functions that are critical and need to be available even when serving a cached page are instead located in bootstrap.inc.

4.2.2 Function Documentation

4.2.2.1 array2object ($ *array*)

Convert an associative array to an anonymous object.

4.2.2.2 check_form ($ *text*)

End of "defgroup search".

4.2.2.3 drupal_access_denied ()

Generates a 403 error if the request is not allowed.

4.2.2.4 drupal_attributes ($ *attributes* = array())

Format an attribute string to insert in a tag.

Parameters:

$attributes An associative array of HTML attributes.

Returns:

An HTML string ready for insertion in a tag.

4.2.2.5 drupal_get_breadcrumb ()

Get the breadcrumb trail for the current page.

4.2.2.6 drupal_get_headers ()

Get the HTTP response headers for the current page.

4.2.2.7 drupal_get_html_head ()

Retrieve output to be displayed in the head tag of the HTML page.

4.2.2.8 drupal_get_normal_path ($ *path*)

Given a path alias, return the internal path it represents.

4.2.2.9 drupal_goto ($ *path* = ”, $ *query* = NULL, $ *fragment* = NULL)

Send the user to a different Drupal page.

This issues an on-site HTTP redirect. The function makes sure the redirected URL is formatted correctly.

It is advised to use drupal_goto() instead of PHP’s header(), because drupal_goto() will append the user’s session ID to the URI when PHP is compiled with "–enable-trans-sid".

This function ends the request; use it rather than a print theme(’page’) statement in your menu callback.

Parameters:

$path A Drupal path.

$query The query string component, if any.

$fragment The destination fragment identifier (named anchor).

4.2.2.10 drupal_http_request ($ *url*, $ *headers* = array(), $ *method* = ’GET’, $ *data* = NULL, $ *retry* = 3)

Perform an HTTP request.

This is a flexible and powerful HTTP client implementation. Correctly handles GET, POST, PUT or any other HTTP requests. Handles redirects.

Parameters:

$url A string containing a fully qualified URI.

$headers An array containing an HTTP header => value pair.

$method A string defining the HTTP request to use.

$data A string containing data to include in the request.

$retry An integer representing how many times to retry the request in case of a redirect.

Returns:

An object containing the HTTP request headers, response code, headers, data, and redirect status.

4.2.2.11 drupal_map_assoc ($ *array*, $ *function* = NULL)

Form an associative array from a linear array.

This function walks through the provided array and constructs an associative array out of it. The keys of the resulting array will be the values of the input array. The values will be the same as the keys unless a function is specified, in which case the output of the function is used for the values instead.

Parameters:

$array A linear array.

$function The name of a function to apply to all values before output.

Returns:
An associative array.

4.2.2.12 drupal_not_found ()

Generates a 404 error if the request can not be handled.

4.2.2.13 drupal_page_footer ()

Perform end-of-request tasks.

This function sets the page cache if appropriate, and allows modules to react to the closing of the page by calling hook_exit().

4.2.2.14 drupal_rebuild_path_map ()

Regenerate the path map from the information in the database.

4.2.2.15 drupal_set_breadcrumb ($ *breadcrumb* = NULL)

Set the breadcrumb trail for the current page.

Parameters:
$breadcrumb Array of links, starting with "home" and proceeding up to but not including the current page.

4.2.2.16 drupal_set_header ($ *header* = NULL)

Set an HTTP response header for the current page.

4.2.2.17 drupal_set_html_head ($ *data* = NULL)

Add output to the head tag of the HTML page.

4.2.2.18 drupal_specialchars ($ *input*, $ *quotes* = ENT_NOQUOTES)

Encode special characters in a string for display as HTML.

Note that we'd like to use htmlspecialchars($input, $quotes, 'utf-8') as outlined in the PHP manual, but we can't because there's a bug in PHP $<$ 4.3 that makes it mess up multibyte charsets if we specify the charset. This will be changed later once we make PHP 4.3 a requirement.

4.2.2.19 drupal_xml_parser_create (&$ *data*)

Prepare a new XML parser.

This is a wrapper around xml_parser_create() which extracts the encoding from the XML data first and sets the output encoding to UTF-8. This function should be used instead of xml_parser_create(), because PHP's XML parser doesn't check the input encoding itself.

This is also where unsupported encodings should be converted. Callers should take this into account: $data might have been changed after the call.

Parameters:

&$data The XML data which will be parsed later.

Returns:

An XML parser object.

4.2.2.20 error_handler ($ *errno*, $ *message*, $ *filename*, $ *line*, $ *variables*)

Log errors as defined by administrator Error levels: 1 = Log errors to database. 2 = Log errors to database and to screen.

4.2.2.21 fix_gpc_magic ()

Correct double-escaping problems caused by "magic quotes" in some PHP installations.

4.2.2.22 l ($ *text*, $ *path*, $ *attributes* = array(), $ *query* = NULL, $ *fragment* = NULL, $ *absolute* = FALSE)

Format an internal Drupal link.

This function correctly handles aliased paths, and allows themes to highlight links to the current page correctly, so all internal links output by modules should be generated by this function if possible.

Parameters:

$text The text to be enclosed with the anchor tag.

$path The Drupal path being linked to, such as "admin/node".

$attributes An associative array of HTML attributes to apply to the anchor tag.

$query A query string to append to the link.

$fragment A fragment identifier (named anchor) to append to the link.

$absolute Whether to force the output to be an absolute link (beginning with http:). Useful for links that will be displayed outside the site, such as in an RSS feed.

Returns:

an HTML string containing a link to the given path.

4.2.2.23 link_node ($ *node*, $ *main* = 0)

Fetch a set of links to display after a given node.

The links are gathered by calls to hook_link('node') in each module.

4.2.2.24 link_page ()

Fetch a set of global navigation links.

The links are gathered by calls to hook_link('page') in each module.

4.2.2.25 locale_initialize ()

Initialize the localization system.

4.2.2.26 message_access ()

Return a string with an "access denied" message.

Always consider whether to use drupal_access_denied() instead to return a proper (and customizable) 403 error.

4.2.2.27 message_na ()

Return a string with a "not applicable" message.

4.2.2.28 object2array ($ *object*)

Convert an object to an associative array.

4.2.2.29 t ($ *string*, $ *args* = 0)

Translate strings to the current locale.

When using t(), try to put entire sentences and strings in one t() call. This makes it easier for translators. HTML markup within translation strings is acceptable, if necessary. The suggested syntax for a link embedded within a translation string is:

```
$msg = t('You must log in below or <a href="%url">create a new
          account</a> before viewing the next page.', array('%url'
          => url('user/register')));
```

We suggest the same syntax for links to other sites. This makes it easy to change link URLs if needed (which happens often) without requiring updates to translations.

Parameters:

$string A string containing the English string to translate.

$args An associative array of replacements to make after translation. Incidences of any key in this array are replaced with the corresponding value.

Returns:
The translated string.

4.2.2.30 url ($ *path* = NULL, $ *query* = NULL, $ *fragment* = NULL, $ *absolute* = FALSE)

Generate an internal Drupal URL.

Parameters:

$path The Drupal path being linked to, such as "admin/node".

$query A query string to append to the link.

$fragment A fragment identifier (named anchor) to append to the link.

$absolute Whether to force the output to be an absolute link (beginning with http:). Useful for links that will be displayed outside the site, such as in an RSS feed.

Returns:
an HTML string containing a link to the given path.

When creating links in modules, consider whether l() could be a better alternative than url().

4.3 conf.php File Reference

Variables

- **$db_url** = "mysql://drupal:drupal@localhost/drupal"
- **$db_prefix** = ""
- **$base_url** = "http://localhost"

4.3.1 Detailed Description

Drupal site-specific configuration file.

4.4 cron.php File Reference

4.4.1 Detailed Description

Handles incoming requests to fire off regularly-scheduled tasks (cron jobs).

4.5 database.inc File Reference

Functions

- db_prefix_tables ($sql)
- db_set_active ($name= 'default')

4.5.1 Detailed Description

Wrapper for database interface code.

4.6 database.mysql.inc File Reference

Functions

- db_connect ($url)
- db_query ($query)
- db_queryd ($query)
- _db_query ($query, $debug=0)
- db_fetch_object ($result)
- db_fetch_array ($result)
- db_num_rows ($result)
- db_result ($result, $row=0)
- db_error ()
- db_next_id ($name)
- db_affected_rows ()
- db_query_range ($query)
- db_encode_blob ($data)
- db_decode_blob ($data)

4.6.1 Detailed Description

Database interface code for MySQL database servers.

4.6.2 Function Documentation

4.6.2.1 _db_query ($ *query*, $ *debug* = 0)

Helper function for db_query().

4.6.2.2 db_affected_rows ()

Determine the number of rows changed by the preceding query.

4.6.2.3 db_decode_blob ($ *data*)

Returns text from a Binary Large OBject value.

Parameters:

$data Data to decode.

Returns:

Decoded data.

4.6.2.4 db_encode_blob ($ *data*)

Returns a properly formatted Binary Large OBject value.

Parameters:
$data Data to encode.

Returns:
Encoded data.

4.6.2.5 db_error ()

Determine whether the previous query caused an error.

4.6.2.6 db_fetch_array ($ *result*)

Fetch one result row from the previous query as an array.

Parameters:
$result A database query result resource, as returned from db_query().

Returns:
An associative array representing the next row of the result. The keys of this object are the names of the table fields selected by the query, and the values are the field values for this result row.

4.6.2.7 db_fetch_object ($ *result*)

Fetch one result row from the previous query as an object.

Parameters:
$result A database query result resource, as returned from db_query().

Returns:
An object representing the next row of the result. The attributes of this object are the table fields selected by the query.

4.6.2.8 db_next_id ($ *name*)

Return a new unique ID in the given sequence.

For compatibility reasons, Drupal does not use auto-numbered fields in its database tables. Instead, this function is used to return a new unique ID of the type requested. If necessary, a new sequence with the given name will be created.

4.6.2.9 db_num_rows ($ *result*)

Determine how many result rows were found by the preceding query.

Parameters:

$result A database query result resource, as returned from db_query().

Returns:

The number of result rows.

4.6.2.10 db_query ($ *query*)

Runs a basic query in the active database.

User-supplied arguments to the query should be passed in as separate parameters so that they can be properly escaped to avoid SQL injection attacks.

Parameters:

$query A string containing an SQL query.

... A variable number of arguments which are substituted into the query using printf() syntax.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

4.6.2.11 db_query_range ($ *query*)

Runs a limited-range query in the active database.

Use this as a substitute for db_query() when a subset of the query is to be returned. User-supplied arguments to the query should be passed in as separate parameters so that they can be properly escaped to avoid SQL injection attacks.

Parameters:

$query A string containing an SQL query.

... A variable number of arguments which are substituted into the query using printf() syntax.

$from The first result row to return.

$count The maximum number of result rows to return.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

4.6.2.12 db_queryd ($ *query*)

Debugging version of db_query().

Echoes the query to the browser.

4.6.2.13 db_result ($ *result*, $ *row* = 0)

Return an individual result field from the previous query.

Only use this function if exactly one field is being selected; otherwise, use db_fetch_object() or db_fetch_array().

Parameters:

$result A database query result resource, as returned from db_query().

$row The index of the row whose result is needed.

Returns:

The resulting field.

4.7 database.pear.inc File Reference

Functions

- db_connect ($url)
- db_query ($query)
- db_queryd ($query)
- _db_query ($query, $debug=0)
- db_fetch_object ($result)
- db_fetch_array ($result)
- db_num_rows ($result)
- db_result ($result, $row=0)
- db_error ()
- db_next_id ($name)
- db_affected_rows ()
- db_query_range ($query)

4.7.1 Detailed Description

Database interface code for database servers using PEAR, including PostgreSQL.

4.7.2 Function Documentation

4.7.2.1 _db_query ($ *query*, $ *debug* = 0)

Helper function for db_query().

4.7.2.2 db_affected_rows ()

Determine the number of rows changed by the preceding query.

4.7.2.3 db_connect ($ *url*)

Initialize a database connection.

4.7.2.4 db_error ()

Determine whether the previous query caused an error.

4.7.2.5 db_fetch_array ($ *result*)

Fetch one result row from the previous query as an array.

Parameters:

$result A database query result resource, as returned from db_query().

Returns:

An associative array representing the next row of the result. The keys of this object are the names of the table fields selected by the query, and the values are the field values for this result row.

4.7.2.6 db_fetch_object ($ *result*)

Fetch one result row from the previous query as an object.

Parameters:

$result A database query result resource, as returned from db_query().

Returns:

An object representing the next row of the result. The attributes of this object are the table fields selected by the query.

4.7.2.7 db_next_id ($ *name*)

Return a new unique ID in the given sequence.

For compatibility reasons, Drupal does not use auto-numbered fields in its database tables. Instead, this function is used to return a new unique ID of the type requested. If necessary, a new sequence with the given name will be created.

4.7.2.8 db_num_rows ($ *result*)

Determine how many result rows were found by the preceding query.

Parameters:

$result A database query result resource, as returned from db_query().

Returns:

The number of result rows.

4.7.2.9 db_query ($ *query*)

Runs a basic query in the active database.

User-supplied arguments to the query should be passed in as separate parameters so that they can be properly escaped to avoid SQL injection attacks.

Parameters:

$query A string containing an SQL query.

... A variable number of arguments which are substituted into the query using printf() syntax.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

4.7.2.10 db_query_range ($ *query*)

Runs a limited-range query in the active database.

Use this as a substitute for db_query() when a subset of the query is to be returned. User-supplied arguments to the query should be passed in as separate parameters so that they can be properly escaped to avoid SQL injection attacks.

Parameters:

$query A string containing an SQL query.

... A variable number of arguments which are substituted into the query using printf() syntax.

$from The first result row to return.

$count The maximum number of result rows to return.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

4.7.2.11 db_queryd ($ *query*)

Debugging version of db_query().

Echoes the query to the browser.

4.7.2.12 db_result ($ *result*, $ *row* = 0)

Return an individual result field from the previous query.

Only use this function if exactly one field is being selected; otherwise, use db_fetch_object() or db_fetch_array().

Parameters:

$result A database query result resource, as returned from db_query().

$row The index of the row whose result is needed.

Returns:

The resulting field.

4.8 database.pgsql.inc File Reference

Functions

- db_query ($query)
- db_queryd ($query)
- _db_query ($query, $debug=0)
- db_fetch_object ($result)
- db_fetch_array ($result)
- db_num_rows ($result)
- db_result ($result, $row=0)
- db_error ()
- db_next_id ($name)
- db_affected_rows ()
- db_query_range ($query)
- db_encode_blob ($data)
- db_decode_blob ($data)

4.8.1 Detailed Description

Database interface code for PostgreSQL database servers.

4.8.2 Function Documentation

4.8.2.1 _db_query ($ *query*, $ *debug* = 0)

Helper function for db_query().

4.8.2.2 db_affected_rows ()

Determine the number of rows changed by the preceding query.

4.8.2.3 db_decode_blob ($ *data*)

Returns text from a Binary Large OBject value.

Parameters:

$data Data to decode.

Returns:

Decoded data.

4.8.2.4 db_encode_blob ($ *data*)

Returns a properly formatted Binary Large OBject value.

Parameters:
$data Data to encode.

Returns:
Encoded data.

4.8.2.5 db_error ()

Determine whether the previous query caused an error.

4.8.2.6 db_fetch_array ($ *result*)

Fetch one result row from the previous query as an array.

Parameters:
$result A database query result resource, as returned from db_query().

Returns:
An associative array representing the next row of the result. The keys of this object are the names of the table fields selected by the query, and the values are the field values for this result row.

4.8.2.7 db_fetch_object ($ *result*)

Fetch one result row from the previous query as an object.

Parameters:
$result A database query result resource, as returned from db_query().

Returns:
An object representing the next row of the result. The attributes of this object are the table fields selected by the query.

4.8.2.8 db_next_id ($ *name*)

Return a new unique ID in the given sequence.

For compatibility reasons, Drupal does not use auto-numbered fields in its database tables. Instead, this function is used to return a new unique ID of the type requested. If necessary, a new sequence with the given name will be created.

4.8.2.9 db_num_rows ($ *result*)

Determine how many result rows were found by the preceding query.

Parameters:

$result A database query result resource, as returned from db_query().

Returns:

The number of result rows.

4.8.2.10 db_query ($ *query*)

Runs a basic query in the active database.

User-supplied arguments to the query should be passed in as separate parameters so that they can be properly escaped to avoid SQL injection attacks.

Parameters:

$query A string containing an SQL query.

... A variable number of arguments which are substituted into the query using printf() syntax.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

4.8.2.11 db_query_range ($ *query*)

Runs a limited-range query in the active database.

Use this as a substitute for db_query() when a subset of the query is to be returned. User-supplied arguments to the query should be passed in as separate parameters so that they can be properly escaped to avoid SQL injection attacks.

Parameters:

$query A string containing an SQL query.

... A variable number of arguments which are substituted into the query using printf() syntax.

$from The first result row to return.

$count The maximum number of result rows to return.

Returns:

A database query result resource, or FALSE if the query was not executed correctly.

4.8.2.12 db_queryd ($ *query*)

Debugging version of db_query().

Echoes the query to the browser.

4.8.2.13 db_result ($ *result*, $ *row* = 0)

Return an individual result field from the previous query.

Only use this function if exactly one field is being selected; otherwise, use db_fetch_object() or db_fetch_array().

Parameters:

$result A database query result resource, as returned from db_query().

$row The index of the row whose result is needed.

Returns:

The resulting field.

4.9 file.inc File Reference

Enumerations

- enum **FILE_DOWNLOADS_PUBLIC**
- enum **FILE_DOWNLOADS_PRIVATE**
- enum **FILE_CREATE_DIRECTORY**
- enum **FILE_MODIFY_PERMISSIONS**
- enum **FILE_DIRECTORY_TEMP**
- enum **FILE_EXISTS_RENAME**
- enum **FILE_EXISTS_REPLACE**
- enum **FILE_EXISTS_ERROR**

Functions

- file_create_url ($path)
- file_create_path ($dest=0)
- file_check_directory (&$directory, $mode=0, $form_item=NULL)
- file_check_path (&$path)
- file_check_upload ($source)
- file_check_location ($source, $directory=0)
- file_copy (&$source, $dest=0, $replace=FILE_EXISTS_RENAME)
- file_move (&$source, $dest=0, $replace=FILE_EXISTS_RENAME)
- **file_create_filename** ($basename, $directory)
- **file_delete** ($path)
- file_save_upload ($source, $dest=0, $replace=FILE_EXISTS_RENAME)
- file_save_data ($data, $dest, $replace=FILE_EXISTS_RENAME)
- file_transfer ($source, $headers)
- file_download ()
- file_scan_directory ($dir, $mask, $nomask=array('.', '..', 'CVS'), $callback=0, $recurse=TRUE)

4.9.1 Detailed Description

API for handling file uploads and server file management.

4.10 index.php File Reference

Variables

- **$status** = menu_execute_active_handler()
- **break**
- case **MENU_ACCESS_DENIED**

4.10.1 Detailed Description

The PHP page that serves all page requests on a Drupal installation.

The routines here dispatch control to the appropriate handler, which then prints the appropriate page.

4.11 locale.inc File Reference

Functions

- _locale_add_language ($code, $name, $onlylanguage=TRUE)
- _locale_admin_manage_screen ()
- _locale_admin_manage_add_screen ()
- _locale_admin_import_screen ()
- _locale_import_po ($file, $lang, $mode)
- _locale_import_read_po ($file)
- _locale_import_parse_header ($header)
- _locale_import_parse_plural_forms ($pluralforms, $filename)
- _locale_import_parse_arithmetic ($string)
- _locale_import_tokenize_formula ($formula)
- _locale_import_append_plural ($entry, $key)
- _locale_import_shorten_comments ($comment)
- _locale_import_parse_quoted ($string)
- _locale_admin_export_screen ()
- _locale_export_po ($language)
- _locale_export_print ($str)
- _locale_export_wrap ($str, $len)
- _locale_export_remove_plural ($entry)
- **_locale_string_delete** ($lid)
- _locale_string_save ($lid)
- _locale_string_edit ($lid)
- _locale_string_language_list ($translation)
- _locale_string_seek_query ()
- _locale_string_seek ()
- _locale_string_seek_form ()
- _locale_prepare_iso_list ()
- _locale_get_iso639_list ()

4.11.1 Detailed Description

Admin-related functions for locale.module.

4.11.2 Function Documentation

4.11.2.1 _locale_add_language ($ *code*, $ *name*, $ *onlylanguage* = TRUE)

Helper function to add a language

4.11.2.2 _locale_admin_export_screen ()

User interface for the translation export screen

4.11.2.3 _locale_admin_import_screen ()

User interface for the translation import screen

4.11.2.4 _locale_admin_manage_add_screen ()

User interface for the language addition screen

4.11.2.5 _locale_admin_manage_screen ()

User interface for the language management screen

4.11.2.6 _locale_export_po ($ *language*)

Exports a Portable Object (Template) file for a language

Parameters:
$language Selects a language to generate the output for

4.11.2.7 _locale_export_print ($ *str*)

Print out a string on multiple lines

4.11.2.8 _locale_export_remove_plural ($ *entry*)

Removes plural index information from a string

4.11.2.9 _locale_export_wrap ($ *str*, $ *len*)

Custom word wrapping for Portable Object (Template) files.

Author:
Jacobo Tarrio

4.11.2.10 _locale_get_iso639_list ()

Some of the common languages with their English and native names

Based on ISO 639 and http://people.w3.org/rishida/names/languages.html

4.11.2.11 _locale_import_append_plural ($ *entry*, $ *key*)

Modify a string to contain proper count indices

This is a callback function used via array_map()

Parameters:

$entry An array element

$key Index of the array element

4.11.2.12 _locale_import_parse_arithmetic ($ *string*)

Parses and sanitizes an arithmetic formula into a PHP expression

While parsing, we ensure, that the operators have the right precedence and associativity.

Parameters:

$string A string containing the arithmetic formula

Returns:

The PHP version of the formula

Author:

Jacobo Tarrio

4.11.2.13 _locale_import_parse_header ($ *header*)

Parses a Gettext Portable Object file header

Parameters:

$header A string containing the complete header

Returns:

An associative array of key-value pairs

Author:

Jacobo Tarrio

4.11.2.14 _locale_import_parse_plural_forms ($ *pluralforms*, $ *filename*)

Parses a Plural-Forms entry from a Gettext Portable Object file header

Parameters:

$pluralforms A string containing the Plural-Forms entry

$filename A string containing the filename

Returns:
An array containing the number of plurals and a formula in PHP for computing the plural form

Author:
Jacobo Tarrio

4.11.2.15 _locale_import_parse_quoted ($ *string*)

Parses a string in quotes

Parameters:
$string A string specified with enclosing quotes

Returns:
The string parsed from inside the quotes

4.11.2.16 _locale_import_po ($ *file*, $ *lang*, $ *mode*)

Parses Gettext Portable Object file information and inserts into database

Parameters:
$file Object contains properties of local file to be imported
$edit Language code
$mode should existing translations be replaced?

4.11.2.17 _locale_import_read_po ($ *file*)

Parses Gettext Portable Object file into an array

Parameters:
$file Object with properties of local file to parse

Author:
Jacobo Tarrio

4.11.2.18 _locale_import_shorten_comments ($ *comment*)

Generate a short, one string version of the passed comment array

Parameters:
$comment An array of strings containing a comment

Returns:
Short one string version of the comment

4.11.2.19 _locale_import_tokenize_formula ($ *formula*)

Backward compatible implementation of token_get_all() for formula parsing

Parameters:
$string A string containing the arithmetic formula

Returns:
The PHP version of the formula

Author:
Gerhard Killesreiter

4.11.2.20 _locale_prepare_iso_list ()

Prepares the language code list for a select form item with only the unsupported ones

4.11.2.21 _locale_string_edit ($ *lid*)

User interface for string editing

4.11.2.22 _locale_string_language_list ($ *translation*)

List languages in search result table

4.11.2.23 _locale_string_save ($ *lid*)

Action handler for string editing

Saves all translations of one string submitted from a form

4.11.2.24 _locale_string_seek ()

Perform a string search and display results in a table

4.11.2.25 _locale_string_seek_form ()

User interface for the string search screen

4.11.2.26 _locale_string_seek_query ()

Build object out of search criteria specified in request variables

4.12 menu.inc File Reference

Menu flags

Flags for use in the "type" attribute of menu items.

- enum **MENU_IS_ROOT**
- enum **MENU_VISIBLE_IN_TREE**
- enum **MENU_VISIBLE_IN_BREADCRUMB**
- enum **MENU_VISIBLE_IF_HAS_CHILDREN**
- enum **MENU_MODIFIABLE_BY_ADMIN**
- enum **MENU_MODIFIED_BY_ADMIN**
- enum **MENU_CREATED_BY_ADMIN**
- enum **MENU_IS_LOCAL_TASK**
- enum **MENU_LINKS_TO_PARENT**

Menu item types

Menu item definitions provide one of these constants, which are shortcuts for combinations of the above flags.

- enum MENU_NORMAL_ITEM
- enum MENU_ITEM_GROUPING
- enum MENU_CALLBACK
- enum MENU_DYNAMIC_ITEM
- enum MENU_SUGGESTED_ITEM
- enum MENU_LOCAL_TASK
- enum MENU_DEFAULT_LOCAL_TASK
- enum MENU_CUSTOM_ITEM
- enum MENU_CUSTOM_MENU

Menu status codes

Status codes for menu callbacks.

- enum **MENU_FOUND**
- enum **MENU_NOT_FOUND**
- enum **MENU_ACCESS_DENIED**

Functions

- menu_get_menu ()
- menu_get_local_tasks ()
- menu_set_location ($location)
- menu_execute_active_handler ()

- menu_get_active_item ()
- menu_set_active_item ($path=NULL)
- menu_get_active_nontask_item ()
- menu_get_active_title ()
- menu_get_active_help ()
- menu_get_active_breadcrumb ()
- menu_in_active_trail ($mid)
- menu_rebuild ()
- theme_menu_tree ($pid=1, $all=FALSE)
- theme_menu_item ($mid)
- theme_menu_local_tasks ()
- theme_menu_local_task ($mid, $active)
- _menu_get_active_trail ()
- _menu_sort ($a, $b)
- _menu_build ()
- _menu_item_is_accessible ($mid)
- _menu_build_visible_tree ($pid=0)
- _menu_append_contextual_items ()
- _menu_find_parents (&$items)
- _menu_build_local_tasks ($pid)

4.12.1 Detailed Description

API for the Drupal menu system.

4.12.2 Function Documentation

4.12.2.1 _menu_append_contextual_items ()

Account for menu items that are only defined at certain paths, so will not be cached.

We don't support the full range of menu item options for these menu items. We don't support MENU_-VISIBLE_IF_HAS_CHILDREN, and we require parent items to be declared before their children.

4.12.2.2 _menu_build ()

Build the menu by querying both modules and the database.

4.12.2.3 _menu_build_local_tasks ($ *pid*)

Find all the items in the current local task tree.

Since this is only for display, we only need title, path, and children for each item.

At the close of this function, $_menu['local tasks'] is populated with the menu items in the local task tree.

Returns:

TRUE if the local task tree is forked. It does not need to be displayed otherwise.

4.12.2.4 _menu_build_visible_tree ($ *pid* = 0)

Find all visible items in the menu tree, for ease in displaying to user.

Since this is only for display, we only need title, path, and children for each item.

4.12.2.5 _menu_find_parents (&$ *items*)

Establish parent-child relationships.

4.12.2.6 _menu_get_active_trail ()

Returns an array with the menu items that lead to the current menu item.

4.12.2.7 _menu_item_is_accessible ($ *mid*)

Determine whether the given menu item is accessible to the current user.

Use this instead of just checking the "access" property of a menu item to properly handle items with fall-through semantics.

4.12.2.8 _menu_sort ($ *a*, $ *b*)

Comparator routine for use in sorting menu items.

4.13 module.inc File Reference

Functions

- module_init ()
- module_iterate ($function, $argument= ”)
- module_list ($refresh=FALSE, $bootstrap=FALSE)
- module_set_filename ($module, $path=null)
- module_get_filename ($module)
- module_get_path ($module)
- module_load ($module)
- module_load_all ()
- module_exist ($module)
- module_hook ($module, $hook)
- module_invoke ($module, $hook, $a1=NULL, $a2=NULL, $a3=NULL, $a4=NULL)
- module_invoke_all ($hook, $a1=NULL, $a2=NULL, $a3=NULL, $a4=NULL)

4.13.1 Detailed Description

API for loading and interacting with Drupal modules.

4.13.2 Function Documentation

4.13.2.1 module_exist ($ *module*)

Determine whether a given module exists.

Parameters:
$module The name of the module (without the .module extension).

Returns:
TRUE if the module is both installed and enabled.

4.13.2.2 module_get_filename ($ *module*)

Retrieve the filename of a module

Parameters:
$module Name of the module which to retrieve the filename of.

Returns:
Filename of module.

4.13.2.3 module_get_path ($ *module*)

Retrieve the path of a module

Parameters:

$module Name of the module which to retrieve the path of.

Returns:

Path of module.

4.13.2.4 module_init ()

Initialize all modules.

4.13.2.5 module_iterate ($ *function*, $ *argument* = ”)

Call a function repeatedly with each module in turn as an argument.

4.13.2.6 module_list ($ *refresh* = FALSE, $ *bootstrap* = FALSE)

Collect a list of all installed and enabled modules.

Parameters:

$refresh Whether to force the module list to be regenerated (such as after the administrator has changed the system settings).

$bootstrap Whether to return the reduced set of modules loaded in "bootstrap mode" for cached pages. See bootstrap.inc.

Returns:

An associative array whose keys and values are the names of all loaded modules.

4.13.2.7 module_load ($ *module*)

Load a module into Drupal, but check first whether a module by the same name has been loaded, and that the filename being included exists.

Parameters:

$module The name of the module to be loaded.

Returns:

TRUE if the load was successful.

4.13.2.8 module_load_all ()

Load all the modules that have been enabled in the system table.

Returns:
TRUE if all modules were loaded successfully

4.13.2.9 module_set_filename ($ *module*, $ *path* = null)

Set the filename of a module, for future loading through module_load()

Parameters:
$module Name of the module which to specify the filename of.

$pa Filename of the module named $module.

Returns:
Filename of module, if no $path has been specified.

4.14 pager.inc File Reference

Pager pieces

Use these pieces to construct your own custom pagers in your theme. Note that you should NOT modify this file to customize your pager.

- pager_first ($text, $limit, $element=0, $attributes=array())
- pager_previous ($text, $limit, $element=0, $interval=1, $attributes=array())
- pager_next ($text, $limit, $element=0, $interval=1, $attributes=array())
- pager_last ($text, $limit, $element=0, $attributes=array())
- pager_detail ($limit, $element=0, $format= '%d through%d of%d.')
- pager_list ($limit, $element=0, $quantity=5, $text= '', $attributes=array())

Functions

- pager_query ($query, $limit=10, $element=0, $count_query=NULL)
- theme_pager ($tags=array(), $limit=10, $element=0, $attributes=array())
- pager_link ($from_new, $element, $attributes=array())
- **pager_load_array** ($value, $element, $old_array)

4.14.1 Detailed Description

Functions to aid in presenting database results as a set of pages.

4.14.2 Function Documentation

4.14.2.1 pager_detail ($ *limit*, $ *element* = 0, $ *format* = '%d through %d of %d.')

Format a summary of the current pager position, such as "6 through 10 of 52".

Parameters:

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$format A printf-style format string for customizing the pager text.

Returns:

An HTML string that generates this piece of the query pager.

4.14.2.2 pager_first ($ *text*, $ *limit*, $ *element* = 0, $ *attributes* = array())

Format a "first page" link.

Parameters:

$text The name (or image) of the link.

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$attributes An associative array of query string parameters to append to the pager links.

Returns:

An HTML string that generates this piece of the query pager.

4.14.2.3 pager_last ($ *text*, $ *limit*, $ *element* = 0, $ *attributes* = array())

Format a "last page" link.

Parameters:

$text The name (or image) of the link.

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$attributes An associative array of query string parameters to append to the pager links.

Returns:

An HTML string that generates this piece of the query pager.

4.14.2.4 pager_link ($ *from_new*, $ *element*, $ *attributes* = array())

Format a link to a specific query result page.

Parameters:

$from_new The first result to display on the linked page.

$element An optional integer to distinguish between multiple pagers on one page.

$attributes An associative array of query string parameters to append to the pager link.

Returns:

An HTML string that generates the link.

4.14.2.5 pager_list ($ *limit*, $ *element* = 0, $ *quantity* = 5, $ *text* = '', $ *attributes* = array())

Format a list of nearby pages with additional query results.

Parameters:

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$quantity The number of pages in the list.

$text A string of text to display before the page list.

$attributes An associative array of query string parameters to append to the pager links.

Returns:

An HTML string that generates this piece of the query pager.

4.14.2.6 pager_next ($ *text*, $ *limit*, $ *element* = 0, $ *interval* = 1, $ *attributes* = array())

Format a "next page" link.

Parameters:

$text The name (or image) of the link.

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$interval The number of pages to move forward when the link is clicked.

$attributes An associative array of query string parameters to append to the pager links.

Returns:

An HTML string that generates this piece of the query pager.

4.14.2.7 pager_previous ($ *text*, $ *limit*, $ *element* = 0, $ *interval* = 1, $ *attributes* = array())

Format a "previous page" link.

Parameters:

$text The name (or image) of the link.

$limit The number of query results to display per page.

$element An optional integer to distinguish between multiple pagers on one page.

$interval The number of pages to move backward when the link is clicked.

$attributes An associative array of query string parameters to append to the pager links.

Returns:

An HTML string that generates this piece of the query pager.

4.15 session.inc File Reference

Functions

- **sess_open** ($save_path, $session_name)
- **sess_close** ()
- **sess_read** ($key)
- **sess_write** ($key, $value)
- **sess_destroy** ($key)
- **sess_gc** ($lifetime)

4.15.1 Detailed Description

User session handling functions.

4.16 tablesort.inc File Reference

Functions

- tablesort_init ($header)
- tablesort_pager ()
- tablesort_sql ($header, $before= ”)
- tablesort_header ($cell, $header, $ts)
- tablesort_cell ($cell, $header, $ts, $i)
- tablesort_get_querystring ()
- tablesort_get_order ($headers)
- tablesort_get_sort ($headers)

4.16.1 Detailed Description

Functions to aid in the creation of sortable tables.

All tables created with a call to theme('table') have the option of having column headers that the user can click on to sort the table by that column.

4.16.2 Function Documentation

4.16.2.1 tablesort_cell ($ *cell*, $ *header*, $ *ts*, $ *i*)

Format a table cell.

Adds a class attribute to all cells in the currently active column.

Parameters:

$cell The cell to format.

$header An array of column headers in the format described in theme_table().

$ts The current table sort context as returned from tablesort_init().

$i The index of the cell's table column.

Returns:

A properly formatted cell, ready for _theme_table_cell().

4.16.2.2 tablesort_get_order ($ *headers*)

Determine the current sort criterion.

Parameters:

$headers An array of column headers in the format described in theme_table().

Returns:
An associative array describing the criterion, containing the keys:

- "name": The localized title of the table column.
- "sql": The name of the database field to sort on.

4.16.2.3 tablesort_get_querystring ()

Compose a query string to append to table sorting requests.

Returns:
A query string that consists of all components of the current page request except for those pertaining to table sorting.

4.16.2.4 tablesort_get_sort ($ *headers*)

Determine the current sort direction.

Parameters:
$headers An array of column headers in the format described in theme_table().

Returns:
The current sort direction ("asc" or "desc").

4.16.2.5 tablesort_header ($ *cell*, $ *header*, $ *ts*)

Format a column header.

If the cell in question is the column header for the current sort criterion, it gets special formatting. All possible sort criteria become links.

Parameters:
$cell The cell to format.

$header An array of column headers in the format described in theme_table().

$ts The current table sort context as returned from tablesort_init().

Returns:
A properly formatted cell, ready for _theme_table_cell().

4.16.2.6 tablesort_init ($ *header*)

Initialize the table sort context.

4.16.2.7 tablesort_pager ()

Fetch pager link arguments.

When producing a sortable table that presents paged data, pass the output of this function into theme('pager') to preserve the current sort order.

4.17 theme.inc File Reference

Functions

- theme_help ($section)
- init_theme ()
- list_themes ($refresh=FALSE)
- list_theme_engines ($refresh=FALSE)
- theme ()
- path_to_theme ()
- theme_get_settings ($key=NULL)
- theme_get_setting ($setting_name, $refresh=FALSE)
- theme_add_style ($style= ”)
- theme_get_styles ()
- theme_page ($content, $title=NULL, $breadcrumb=NULL)
- theme_status_messages ()
- theme_links ($links, $delimiter= ’| ’)
- theme_image ($path, $alt= ”, $title= ”, $attr= ”, $getsize=true)
- theme_breadcrumb ($breadcrumb)
- theme_node ($node, $teaser=FALSE, $page=FALSE)
- theme_form_element ($title, $value, $description=NULL, $id=NULL, $required=FALSE, $error=FALSE)
- theme_submenu ($links)
- theme_table ($header, $rows, $attributes=NULL)
- theme_box ($title, $content, $region= ’main’)
- theme_block ($block)
- theme_mark ()
- theme_stylesheet_import ($stylesheet, $media= ’all’)
- theme_item_list ($items=array(), $title=NULL)
- theme_error ($message)
- **theme_more_help_link** ($url)
- theme_xml_icon ($url)
- theme_closure ($main=0)
- theme_onload_attribute ($theme_onloads=array())
- theme_blocks ($region)
- _theme_table_cell ($cell, $header=0)

4.17.1 Detailed Description

The theme system, which controls the output of Drupal.

The theme system allows for nearly all output of the Drupal system to be customized by user themes.

See also:

Theme system
themeable

4.17.2 Function Documentation

4.17.2.1 _theme_table_cell ($ *cell*, $ *header* = 0)

End of "defgroup themeable".

4.17.2.2 init_theme ()

Initialize the theme system by loading the theme.

Returns:
The name of the currently selected theme.

4.17.2.3 list_theme_engines ($ *refresh* = FALSE)

Provides a list of currently available theme engines

Parameters:
$refresh Whether to reload the list of themes from the database.

Returns:
An array of the currently available theme engines.

4.17.2.4 list_themes ($ *refresh* = FALSE)

Provides a list of currently available themes.

Parameters:
$refresh Whether to reload the list of themes from the database.

Returns:
An array of the currently available themes.

4.17.2.5 path_to_theme ()

Return the path to the currently selected theme.

4.17.2.6 theme ()

Generate the themed representation of a Drupal object.

All requests for themed functions must go through this function. It examines the request and routes it to the appropriate theme function. If the current theme does not implement the requested function, then the current theme engine is checked. If neither the engine nor theme implement the requested function, then the base theme function is called.

For example, to retrieve the HTML that is output by theme_page($output), a module should call theme('page', $output).

Parameters:

$function The name of the theme function to call.

... Additional arguments to pass along to the theme function.

Returns:

An HTML string that generates the themed output.

4.17.2.7 theme_add_style ($ *style* = '')

Add a theme stylesheet to be included later. This is handled separately from drupal_set_html_head() to enforce the correct CSS cascading order.

4.17.2.8 theme_get_setting ($ *setting_name*, $ *refresh* = FALSE)

Retrieve a setting for the current theme. This function is designed for use from within themes & engines to determine theme settings made in the admin interface.

Caches values for speed (use $refresh = TRUE to refresh cache)

Parameters:

$setting_name The name of the setting to be retrieved.

$refresh Whether to reload the cache of settings.

Returns:

The value of the requested setting, NULL if the setting does not exist.

4.17.2.9 theme_get_settings ($ *key* = NULL)

Retrieve an associative array containing the settings for a theme.

The final settings are arrived at by merging the default settings, the site-wide settings, and the settings defined for the specific theme. If no $key was specified, only the site-wide theme defaults are retrieved.

The default values for each of settings are also defined in this function. To add new settings, add their default values here, and then add form elements to system_theme_settings() in system.module.

Parameters:

$key The template/style value for a given theme.

Returns:
An associative array containing theme settings.

4.17.2.10 theme_get_styles ()

Return the HTML for a theme's stylesheets.

4.17.2.11 theme_help ($ *section*)

Hook Help - returns theme specific help and information.

Parameters:
section defines the *section* of the help to be returned.

Returns:
a string containing the help output.

4.18 update.php File Reference

Functions

- **update_data** ($start)
- **update_page_header** ($title)
- **update_page_footer** ()
- **update_page** ()
- **update_info** ()

Variables

- **$access_check** = TRUE

4.18.1 Detailed Description

Administrative page for handling updates from one Drupal version to another.

Point your browser to "http://www.site.com/update.php" and follow the instructions.

If you are not logged in as administrator, you will need to modify the access check statement below. Change the TRUE into a FALSE to disable the access check. After finishing the upgrade, be sure to open this file and change the FALSE back into a TRUE!

4.19 xmlrpc.php File Reference

Variables

- **$functions** = module_invoke_all(xmlrpc)
- **$server** = new xmlrpc_server($functions)

4.19.1 Detailed Description

PHP page for handling incoming XML-RPC requests from clients.

Index