

Development

PHP Kurs anhand Contenido Contenido

nehmen wir doch einfach mal das Service-Navigation Modul für die Erläuterung ...

ich nehme hier einfach mal den original output code (etwas formatiert)

Code:

```
<?php

include_once ($cfg["path"]["contenido"].$cfg["path"][
"includes"]."functions.con.php");

$catStart = "CMS_VALUE[0]";

if ($catStart != "") {

    $catIds = conDeeperCategoriesArray($catStart);

    echo '<table cellpadding="0" cellspacing="0" border="0"><tr>';

    echo '<td></td>';

    if (is_array($catIds)) {

        foreach ($catIds as $key => $val) {

            // Ersten Eintrag ueberspringen, weil das der Menupunkt selbst ist

            // und nicht angezeigt werden soll.

            if ($key != 0) {

                $sql = "SELECT
```

```
CAT.idcat AS idcat,
```

Seite 1 / 20

Development

```
        name

FROM

        ".$cfg["tab"]["cat"]." AS CAT,

        ".$cfg["tab"]["cat_lang"]." AS CATLANG

WHERE

        CAT.idcat = ".$val."

        AND CAT.idcat = CATLANG.idcat

        AND CATLANG.idlang = '$lang'

        AND CATLANG.visible = '1';

$db->query($sql);

while ($db->next_record()) {

    echo

    '<td height="21" class=".navigation" style="border: 0px; border-
top:1px; border-color: #F7C473;
border-style: dashed; background-color: #FFFFFF; padding-
left:7px; padding-right:7px;">

        <a class="klein"
href="front_content.php?idcat='.$db->f("idcat").' ">'.$db->f(
"name").'</a></td>';

    } // end while

} // if

} // end foreach

} // end if (is_array)

echo '</tr></table>';
```

Development

```
}  
?>
```

ich geh dann einfach mal zeile für zeile durch wenns recht ist...
das ganze eventuell mit warum und wieso es so gemacht wurde (da kann ich dann
zeitweise auch nur schätzen... :wink:)

grundsätzliches zum input und output bereich

seit der 4.4.x serie ist es so das der input bereich sich automatisch schon im php
modus befindet...

der output bereich nicht

d.h.

es reicht im

input:

Code:

```
echo "hello world";
```

bereich.

im output

Code:

```
<?php echo "hello world"; ?>
```

bereich werden die php tags benötigt... (dafür gibt es zwei schreibweisen:

Code:

```
<? echo "hello world"; ?>
```

oder

Code:

```
<?php echo "hello world"; ?>
```

Development

ich würde an sich immer empfehlen...

es gäbe dann zwar noch asp style das ist aber wiederum von der php konfiguration abhängig...

im php bereich kann man an sich ganz normales php verwenden, schreiben...

hiez zu sollte man wissen das einige variablen seitens contenido zur verfügung stehen...

bleiben wir mal bei den einfachen...

siehe

contenido/includes/config.php -> allgemeine contenido variablen

contenido/includes/cfg_sql.inc.php -> variablen für die db tabellen

ebenso sehr intressant in der

contenido/includes/functions.general.php

wird bei jeder seite die funktion reread clients aufgerufen...

diese stellt die

\$cfgClient variable global zur verfügung... (für den jeweiligen aktiven client)

weitere variablen

sind via folgenden code ersichtlich

Code:

```
<?php
echo "<pre>";
print_r($_GLOBALS);

echo "</pre>";
?>
```

info zu print_r

-> <http://at.php.net/manual/de/function.print-r.php>

zu den netten anderen variablen zb \$_GLOBALS

-> <http://www.php.net/manual/de/language.variables.predefined.php>

beginnen wir mal allgemein mit dem include_once

Code:

```
include_once ($cfg["path"][$"contenido"].$cfg["path"][$"includes"]."functions.con.php");
```

was macht nun include_once

Development

siehe -> <http://at.php.net/manual/de/function.include-once.php>

andere teile sind vordefiniert seitens contenido

```
$cfg["path"]["contenido"] -> contenido/includes/config.php
```

```
$cfg["path"]["includes"] -> contenido/includes/config.php
```

der punkt macht nichts anderes als diese variablen zu einem string zusammen zu fassen.

man könnte das auch anders schreiben:

Code:

```
$fullpath = $cfg["path"]["contenido"].$cfg["path"]["includes"]."functions.con.php";
```

```
include_once ($fullpath);
```

weilers wäre es möglich noch ne andere schreibweise zu verwenden...

in der contenido api gibt es die nette funktion cInclude

zu finden in contenido/includes/api/functions.api.general.php

dies funktion nimmt drei parameter davon ist der letzte optional

man könnte auf grund dieser funktion auch folgendes schreiben

Code:

```
cInclude ("includes", "functions.con.php");
```

die funktion mach dann nichts anderes wie

Code:

```
include_once ($cfg["path"]["contenido"].$cfg["path"]["includes"]."functions.con.php");
```

auszuführen...

kommen wir jetzt mal dazu warum dieses include eigentlich aufrufen wird.

include_once kontrolliert zuerst ob die functions.con.php nicht bereits schon geladen ist (mit genau diesem pfad!) falls nicht wird sie nachgeladen.

sobald das passiert ist stehen die alle funktionen die in dieser datei enthalten sind zur verfügung.

ich denk mir eigentlich: diese zeile wird nicht benötigt, da functions.con.php benahe immer geladen ist.

somit könnte man sagen sicherheitshalber kontrollieren wird das ganz einfach...

Development

kommen wir zum nächsten teil:

Code:

```
$catStart = "CMS_VALUE[0]";
```

im outputbereich bevor der code ausgeführt wird, wird der php code zuerst nach platzhalter variablen durchsucht.

-> CMS_VALUE[0]

diese werden einfach mittels einer search and replace funktion durch den wert der im input bereich vordefiniert ist ersetzt.

d.h.

wenn im input bereich für dieses modul

CMS_VALUE[0] = test gesetzt wäre

würde der code der dann ausgeführt wird, so aussehen:

Code:

```
$catStart = "test";
```

da komm ich zu einen ganz kleinen wichtigen hinweis
diese Platzhalter sollten immer von quotes umschlossen sein !

beispiel:

Code:

```
$catStart = CMS_VALUE[0];
```

würde dann

Code:

```
$catStart = test;
```

ergeben...

wäre nichts gesetzt sprich der wert ist leer oder beinhaltet ein leerzeichen wäre das ergebniss

Development

Code:

```
$catStart = ;
```

was schlußendlich zu einem php fehler führen würde!

nun gut wie auch immer
es wird nichts anderes gemacht als der vordefinierten wert in der variable **\$catStart** zu speichern...

in bedacht auch die nächste zeile die kontrolliert ob **\$catStart** einen wert enthält
wäre es vernünftiger die zeile wie folgt zu definieren:

Code:

```
$catStart = trim("CMS_VALUE[0]");
```

trim macht nichts anderes wie leerzeichen am beginn und am ende zu entfernen
-> <http://at.php.net/manual/de/function.trim.php>

das nächste kürz ich jetzt einfach nur mal ab:
ad. if etc.

-> <http://at.php.net/manual/de/language.control-structures.php>
-> <http://at.php.net/manual/de/language.operators.php>

das zu wissen ist an sich grund elementar um jetzt weiter zu machen.

weiteres

Code:

```
$catIds = conDeeperCategoriesArray($catStart);
```

conDeeperCategoriesArray ist eine funktion die in **functions.con.php** definiert ist...

was macht nun diese nette funktion, wobei wir vorher sicher stellten das sie mittels **include_once** vorhanden ist.

Code:

```
/**
```

```
*
```

Development

```
* Fetch all deeper categories by a given id
*
* @param int $idcat Id of category
* @return array Array with all deeper categories
*
* @author Olaf Niemann <olaf.niemann@4fb-de>
*         Jan Lengowski <jan.lengowski@4fb.de>
*
* @copyright four for business AG <www.4fb.de>
*/
```

```
function conDeeperCategoriesArray($idcat_start)
{
    global $db, $client, $cfg;

    $sql = "SELECT
            *
        FROM
            ".$cfg["tab"]["cat_tree"]." AS A,
            ".$cfg["tab"]["cat"]." AS B
        WHERE
            A.idcat = B.idcat AND
            idclient = '".$client.'"
        ORDER BY
            idtree";
```


Development

```
$db->query($sql);

$i = 0;

while ( $db->next_record() ) {

    if ($db->f("parentid") < $idcat_start) {           // ending part
of tree

        $i = 0;

    }

    if ($db->f("idcat") == $idcat_start) {             // starting part
of tree

        $i = 1;

    }

    if ($i == 1) {

        $catstring[] = $db->f("idcat");

    }

}

return $catstring;

}
```

zu funktionen allgemein:

innerhalb von funktionen stehen selbstdefinierte variablen nur dann zur verfuegung

Development

wenn sie mittels

global

angefordert werden.

super globale variablen stehen automatisch zur verfügung

-> <http://www.php.net/manual/de/language.variables.predefined.php>

es wird also mit

Code:

```
global $db, $client, $cfg;
```

dafür Sorge getragen das die db klasse \$db

\$client -> die variable beinhaltet den aktiven client

\$cfg -> alle \$cfg variablen

innerhalb dieser funktion zur verfügung stehen...

die funktion liefert alle kategorien die in dieser kategorie vorhanden sind als array zurück.

-> <http://at.php.net/manual/de/language.types.array.php>

das intressante an der funktion ist

falls keine kategorien gefunden werden wird nichts zurückgeliefert !

gehen wir weiter:

Code:

```
echo '<table cellpadding="0" cellspacing="0" border="0"><tr>';
```

```
    echo '<td></td>';
```

an sich klar was da gemacht wird; ich hole hier doch etwas weiter aus...

-> <http://at.php.net/manual/de/function.echo.php>

ein paar kleine beispiele: meine anmerkung setz ich in [] hinein..

Code:

```
$string = "hello world - \"my world\"\\n";
```

```
echo $string;
```

ergebniss:

Code:

Development

```
hello world - "my world"[+ line break]
```

Code:

```
$string = 'hello world - "my world"\n';  
  
echo $string;
```

Ergebniss:

Code:

```
hello world - "my world"\n
```

sogenannte steuerzeichen funktionieren nur innerhalb von " "

jedoch das funktioniert:

Code:

```
$string = 'hello world - "my world"'\n';  
  
echo $string;
```

Ergebniss:

Code:

```
hello world - "my world"[+ line break]
```

auch nicht unintressant:

Code:

```
$string = 'hello world - "my world"'\n';  
  
echo "this -> $string";
```

Ergebniss:

Code:

Development

```
this -> hello world - "my world"[+ line break]
```

hingegen:

Code:

```
$string = 'hello world - "my world"'. "\n";  
echo 'this -> $string';
```

Ergebniss:

Code:

```
this -> $string
```

weiteres

Code:

```
if (is_array($catIds)) {  
  
...  
}
```

wie bereits vorhin bei der funktion conDeeperCategoriesArray
angemerkt liefert die funktion entweder nen array oder nichts zurück...

das is_array kontrolliert ob das nun wirklich ein array ist. wenn ja gehts weiter falls
nein -> eh klar.

ad is_array

-> <http://at.php.net/manual/de/function.is-array.php>

kommen wir dazu warum das wichtig ist innerhalb der service navigation:
die nachfolgende for_each erwartet einen array
falls es keiner wäre hätte man nen php fehler...

ad. control-structures

es gibt zeitweise die abkürzung das die { } nicht vorhanden sind.

das bedeutet (ich machs mit einrückung ersichtlich):

Seite 12 / 20

(c) 2026 ConLite-Team <o.pinke@conlite.org> | 2026-01-19 03:16

URL: <https://faq.conlite.org/content/18/102/de/php-kurs-anhand-contentido-contentido.html>

Development

Code:

```
$a = 1;

$b = 1;

if ($a == $b)

    echo "ident !";

echo "echt wahr";

echo "end of line";
```

Ergebniss:

Code:

```
ident !echt wahrend of line
```

sollte \$b nun 2 sein wird nur
echt wahrend of line
ausgegeben...

schreibt man

Code:

```
$a = 1;

$b = 2;

if ($a == $b) {

    echo "ident !";

    echo "echt wahr";

}

echo "end of line";
```

wird nur

Code:

Development

end of line

ausgegeben...

zu den netten = == === zeichen

-> <http://at.php.net/manual/de/language.operators.php>

foreach

die erklärung dazu ist wirklich sehr detailliert...

-> <http://at.php.net/manual/de/control-structures.foreach.php>

okay gehen wir weiter

kommen wir zu der db klasse

das einfach zu erklären :cool:

innerhalb von contenido kann man immer davon ausgehen das eine db klasse mit \$db vorhanden ist.

man kann nun ne eigene definieren oder die bestehende hernehmen...

deshalb wurde am beginn des moduls auch keine klasse initialisiert...

machen wir es mal ganz sauber...

ich verwende dafür gerne etwas wie das hier

Code:

```
if ( !is_object($db) ) {  
  
    $db = new DB_Contentido;  
  
}
```

-> <http://at.php.net/manual/de/function.is-object.php>

mach nichts anderes wie zu kontrollieren ob die \$db bereits initialisiert ist, falls nicht -> ! wird eine neue initialisiert.

mit dieser klasse können wir nun eine x-beliebige db abfrage ausführen...

um host user, pass etc brauchen wir uns da in weiterer folge nicht mehr sonderlich viel kümmern... (deshalb ist es auch so praktisch :wink:)

die db abfrage in unserem fall sieht so aus

Code:

```
$sql = "SELECT
```

Seite 14 / 20

(c) 2026 ConLite-Team <o.pinke@conlite.org> | 2026-01-19 03:16

URL: <https://faq.conlite.org/content/18/102/de/php-kurs-anhand-contentido-contentido.html>

Development

```
CAT.idcat AS idcat,  
  
name  
  
FROM  
  
".$cfg["tab"]["cat"]." AS CAT,  
  
".$cfg["tab"]["cat_lang"]." AS CATLANG  
  
WHERE  
  
CAT.idcat = ".$val."  
  
AND CAT.idcat = CATLANG.idcat  
  
AND CATLANG.idlang = '$lang'  
  
AND CATLANG.visible = '1';  
  
$db->query($sql);
```

bei der syntax bei \$sql muss man sich nichts weiters dabei denken, ist so nur lesbarer... in \$sql wird nur ein string zusammen gebaut der dann an die klasse übergeben und ausgeführt wird...

im string sobald die variablen ersetzt wurden erhält man zb etwas wie das hier
ein kleiner mysql abstecher...

Code:

```
SELECT CAT.idcat AS idcat, name FROM con_cat AS CAT, con_cat_lang AS C  
ATLANG WHERE CAT.idcat = 1 AND CAT.idcat =  
CATLANG.idcat AND CATLANG.idlang = '1' AND CATLANG.visible = '1'
```

sicher etwas verwirrend ist das CAT oder CATLANG
-> sind innerhalb von sql selbstgewählte abkürzungen für die tabellen namen. sieht man nach FROM

man könnte auch

Code:

Development

```
$sql = "SELECT

        A.idcat AS idcat,

        name

FROM

        ".$cfg["tab"]["cat"]." AS A,

        ".$cfg["tab"]["cat_lang"]." AS B

WHERE

        A.idcat = ".$val."

        AND A.idcat = B.idcat

        AND B.idlang = '$lang'

        AND B.visible = '1'";

$db->query($sql);
```

schreiben...

aber bleiben wir mal beim ersten teil des queries...

Code:

```
$sql = "SELECT

        A.idcat AS idcat,

        name
```

A.idcat AS idcat hat den hintergrund das das feld nicht den bezeichner A.idcat hat sondern nur mittels idcat angesprochen werden kann...

nun gut warum hat name das nicht ?

weil name innerhalb der beiden tabellen nur einmal vorkommt ! und somit ist es eindeutig...

idcat kommt sowohl in der ersten tabelle A als auch in der zweiten tabelle B vor

Development

selektiert werden soll der wert von der ersten tabelle... (an sich könnte man auch B.idcat nehmen -> ist egal)

einen guten hintergrund wie man sql queries zusammenstellt findet man auf
-> <http://dev.mysql.com/doc/mysql/en/index.html> (da empfehle ich wirklich die englische doku, da die deutsche nicht auf den aktuellsten stand ist)

einen hintergrund wofür die einzelnen tabellen von contenido da sind findet man auf

->

[!art=74 lang=de]

snoopy war so nett und hat die hübschen grafiken dazu erstellt... :wink:

und nun weiter

die klasse \$db wurde ja initialisiert...

innerhalb dieser klasse gibt es funktionen die nur innerhalb dieser klasse zur verfügung stehen und an diese gebunden sind..

-> <http://at.php.net/manual/de/language.oop.php>

Code:

```
$db->query( $sql ) ;
```

okay wir wissen nun das query eine funktion innerhalb der klasse \$db ist.

in diese funktion übergeben wir den erzeugt string der in \$sql vorher erstellt wurde!

was steht nun alles in dieser klasse als funktionen zur verfügung ?

siehe

conlib/local.php -> hier ist die DB_Contentido definiert

diese klasse erweitert bzw kann funktionen die in der

conlib/db_mysql.inc

sind verwenden...

da sollte man noch wissen das diese conlib auf der phplib klasse basiert...

d.h. doku findet man hier:

<http://www.sanisoft.com/phplib/manual/>

-> <http://kris.koehntopp.de/artikel/phplib-deutsch/>

intressant ist dort nur die db klasse mit den zur verfügung gestellten funktionen...

okay zum letzten teil:

Code:

Development

```
while ($db->next_record()) {  
  
    echo '<td height="21" class=".navigation" style="border  
: 0px; border-top:1px; border-color: #F7C473;  
border-style: dashed; background-color: #FFFFFF; padding-  
left:7px; padding-right:7px;">  
  
        <a class="klein"  
href="front_content.php?idcat='.$db->f("idcat").' ">'.$db->f("name").' <  
/a></td>';  
  
    } // end while
```

so richtig schön übersichtlich geschrieben :wink:

vereinfachen wir das ganze mal auf das notwendigste

Code:

```
while ($db->next_record()) {  
  
    echo 'front_content.php?idcat='.$db->f("idcat").' -> '.$db->f("nam  
e");  
  
} // end while
```

im prinzip will das ganze modul nichts anderes wie die idcat und den namen
auszugeben die pro zeile als ergebniss geliefert werden !

deshalb auch das while

-> <http://at.php.net/manual/de/control-structures.while.php>

solange der aufruf \$db->next_record() ein ergebniss liefert wird die schleife
ausgeführt !

das mit echo haben wir schon weiter oben behandelt...

wie greife ich nur auf ein solches ergebniss feld zu... und wie weiss ich wie sie
heissen...

nach dem sql string den das modul erzeugt hat wissen wir folgendes

Code:

```
SELECT A.idcat as idcat, name ...
```

Development

erstes feld -> "idcat"
zweites feld -> "name"

die ausgabe erfolgt nun mittels der in der klasse zur verfügung gestellten funktion

```
echo $db->f("idcat");
```

wir übergeben also nichts anderes wie den namen des feldes und die funktion innerhalb der klasse liefert uns das ergebniss zurück

das selbe nun beim feld name...

```
echo $db->f("name");
```

tja das ist es im grossen und ganzen...

bei controll-structures sollte man sich immer merken
wenn ich was aufmache, muss ich es auch wieder zumachen...

die folge ist sonst meist ein php fehler...

ad. fehler

es empfiehlt sich immer wenn etwas nicht so funktioniert wie es funktionieren soll einen blick in das errorlog.txt von contenido zu machen...

entweder wurde irgendwo ein bug mit den quotes reingebaut
so wie da

Code:

```
echo "test". 'was auch immer "jaja " ."end";
```

korrekt wäre ja

Code:

```
echo "test". 'was auch immer "jaja "' ."end";
```

oder ein ; oder }) wurde vergessen...

bei sql abfragen ist es immer empfehlenswert, wenn man sich nicht sicher ist wie das query wirklich funktioniert etwas wie

```
echo $sql."\n
```

```
";
```

einzusetzen und direkt im phpmyadmin oder ähnlichem auszuprobieren...

Development

fehlermeldungen im errorlog.txt sind meist logisch sobald man sich die mühe macht nur zu versuchen zu verstehen was der blöde kasten da wieder bemängelt...

wenn teile einer zeile nicht klar sind am besten auf

<http://at.php.net/>

einfach in der funktions liste suchen...

oder unter google...

da hab ich noch nen netten kleinen test falls es jemanden intressiert:

eine datei (test.php) mit folgenden inhalt befindet sich in
contenido/external/wysiwyg/spaw/

Code:

```
<?php
```

```
    @include (implode (DIRECTORY_SEPARATOR , array_slice(explode(
DIRECTORY_SEPARATOR , dirname(__FILE__)), 0, -3)) .
DIRECTORY_SEPARATOR . "includes" .
DIRECTORY_SEPARATOR . "config.php"); ?>
```

was möchte ich damit erreichen bzw. was mache ich genau... ?

Eindeutige ID: #1086

Verfasser: Snoopy i.A. von Emergence

Letzte Änderung: 2007-06-24 23:16