开发、把玩 drupal 指引

Drupal 拥有极佳的弹性，允许各种类型的应用，然而在开始动手前，总是得先了解一下 Drupal 的核心运作原理，如此以来才会得心应手。

下面搜集了一些把玩 Drupal 的范例和教学，如果你觉得这里无法满足你的需求，还可以至 drupal.org 瞧瞧，找找看是否有对你有用的资讯。

Drupal API：提供 Drupal 核心函式的范例和说明 http://drupaldocs.org/api

Development for Drupal：官方为开发者提供详尽的说明文件 http://drupal.org/contributors-guide

我是初学者，想要自行开发该如何上手

大家好，我是最近才接触 CMS 的新手，在试着安装过 joomla、e107、xoops 之后，最后透过友人的推荐，决定使用 drupal 来架设网站，目前网站已经架设完成，但是非常阳春，我还是个初学者，其实很多东西也不是了解，市面上好像也没有 drupal 的书，可是我想要自己开发新的模组(或修改现成的模组)建立自己的网站，我该从何上手呢，有没有什么资料值得我先阅读的，我想从基本学起，也许只是从自己写一个功能简单的搜寻关键字的模组开始，慢慢渐入佳境，可是目前对 drupal 架构完全没什么概念，光是看原始码我就有点头痛了，所以想自己写一些简单基本的功能，有没有人可以给我一些意见要如何上手。谢谢

Code Snippets：计算 Views 生成资料的总数

Views 用来「列资料」很棒，可是如果要做一些统计、计算，虽然有些模组可以用（像是 Views Cul 等等），但其实都不是很好的解决方案。不过 Views 的 API 非常齐全，所以只要了解 Views 的写法，也是可以不需要自己写 query 喔！

以下范例是算出某个 Views 的 total rows，也就是该 Views 的总数。
实际应用举例：
奖金猎人的「目前进行中比赛有 XXX 件」这样的区块。

```
<?php
$view = views_get_view ( 'VIEWS_NAME' );
$view -> get_total_rows = TRUE ;
$view -> execute ();
$count = $view -> total_rows ; print
```

```
'目前进行中比赛有' . $count . 'XXX 件' ;
?>
```

如果你的状况需要设参数的话：

```
<?php
$view = views_get_view ( 'VIEWS_NAME' );
$view -> set_arguments ( array( 1 , 2 , 3 ) ); //参数设在这
$view -> get_total_rows = TRUE ;
$view -> execute () ;
$count = $view -> total_rows ; print

$count ;
?>
```

如果希望最后输出可以依照总数改变的话：

```
<?php
$view = views_get_view ( 'VIEWS_NAME' );
$view -> set_arguments ( array( 1 , 2 , 3 ) ); //参数设在这
$view -> get_total_rows = TRUE ;
$view -> execute () ;
$count = $view -> total_rows ;
if ( $count > 0 ) {
  $output = format_plural ( $count ,
    '只有 1 个' ,
    '总共有@count 个' );
  print $output ;
}
?>
```

drupal_foo

drupal_ foo 的意思，就是 drupal 很多函数里面以「drupal_」为开头名称的，会以 drupal 开头的函数，表示很常被模组的开发者使用，也特别的重要。

几乎所有的 drupal_foo 函数，都会在下面这些位置找到。

include/bootstrap.inc
include/common.inc

也建议模组开发者，可以善加利用 drupal 核心提供的 API，并注意不要自行在模组里新增 drupal_foo 的 function，以免混淆。

> 这个单元是许多热心的贡献者，将许多 drupal_ foo 函数写上中文说明，每个 drupal_foo 的文件说明里里通常会包含：定义、描述、参数、传回值、范例、程式码...等等。若您也知道一些 drupal_foo 的用途，或是看到有错误的地方，欢迎直接共笔编辑，帮忙加上说明和解释。

drupal_add_js

定义

drupal_add_js($data = NULL, $type = 'module', $scope = 'header', $defer = FALSE, $cache = TRUE)
includes/common.inc,第 1600 行之后

描述

加入一个 JavaScript 的档案、设定、或是程式码到页面中

根据不同的参数，会有不同的结果产生。一般，它会加入 JavaScript 的"档案连结"或是"直接贴入程式码"到网页里。以下列出不同的结果：

- 加入一个档案('core', 'module' and 'theme'):加入一个连结到网页中。排列的顺有一定的规则，'core'最前面，再来是'module' ，最后是'theme'，就算加入的时间比较晚，还是会依这个顺序排列。
  例：
  drupal_addjs('js1.js', 'module');
  drupal_addjs('js2.js', 'core');

  因为'core' 的顺序在前面，所以 js2.js 会排在 js1.js 前面，会优先被读入。
- 加入一段程式码('inline'): 贴入程式码，如果目的不是引用一个*.js 的档案，而是一段 JavaScript 程式码，就用这个参数。例如，开一个新视窗，或是跳出讯息视窗显示一段讯息。
- * 加入一个设定('setting'): 传入一个变数的值。这个值会被放到名为 Drupal.settings 的阵列中，让其它部分的 JavaScript 去呼叫它。

参数

$data (选择性) 如果用到这个参数，它的内容的和后面的$type 参数有关：

- 'core', 'module' 或'theme': 一个由网站根目录算起的相对路径(*.js 档案)
- 'inline': 一段 JavaScript 程式码
- 'setting': 一个关连阵列，将会被转换为 Drupal.settings 下的物件

$type (选择性)一个表示要放到网页里的 JavaScript 是哪一种类型。可以是'core', 'module', 'theme', 'inline'和'setting'。你也可以自订一个名称，这时$data 的内容会被视为一个 JavaScript 档案(*.js)路径。预设值是'module' 排列顺序： 'core' -> 'module' -> 'theme' -> 'setting' -> 'inline' -> [自订的名称]

$scope (选择性) JavaScript 程式要放置的位置，预设可以是'header'(开头) and 'footer'(节尾). 如果你所用的版型有定义其它的位置，也可以使用它们放到这个参数中。

$defer (选择性) 如果为 TRUE，在标签中会加入 defer 属性，预设为 FALSE。这个参数在$type 是'setting' 没有作用。

$cache (选择性) 如果设定为 FALSE， JavaScript 档案会在每一次呼叫页面的时候重新读取，预设值是 TRUE，这个参数只对$type 表示引入的是一个 JavaScript 档案有用

值回值

如果第一个参数为 NULL，就会传回和$scope 同样参数 JavaScript 阵列回来

范例

1. 引入档案

```
drupal_add_js('test1.js', 'module');
drupal_add_js('test2.js', 'theme');
drupal_add_js('test3.js', 'core');
```

结果网页原始档的开头部分可以找到类似下面的程式码

```
<script type="text/javascript" src="/drupal/test3.js"></script>
<script type="text/javascript" src="/drupal/test1.js"></script>
<script type="text/javascript" src="/drupal/test2.js"></script>
```

由于 core 的较优先，所以会被排在前面先被读入。

2. 加入程式码& 加入设定

```
drupal_add_js('alert("var1 = "+Drupal.settings.var1);', 'inline');
drupal_add_js(array('var1' => 123, 'var2' => 456), 'setting');
```

执行的时候，会跳出一个讯息视窗，显示「var1 = 123」。

原始码

```
<?php
function drupal_add_js ( $data = NULL , $type = 'module' , $scope = 'header' , $defer = FA
LSE , $cache = TRUE ) {
  if (! is_null ( $data )) {
    _drupal_add_js ( 'misc /jquery.js' , 'core' , 'header' , FALSE , $cache );
    _drupal_add_js ( 'misc/drupal.js' , 'core' , 'header' , FALSE , $cache );
  }
  return _drupal_add_js ( $data , $type , $scope , $defer , $cache );
}
?>
```

drupal_get_path

定义

drupal_get_path($type, $name)

includes/common.inc

描述

可用此函数取得模组、版型或版型引擎(theme engine)的路径

参数

$type:要寻找的种类(如： theme, theme_engine, module)

$name:寻找东西的名字

传回值

要找东西的路径(网站相对路径，不是伺服器的系统路径)

范例

当你开发的模组里面有图片，或是由其它需要引入的 JavaScript 的时候，那就很有用了。因为你不能确定你的模组会被放在哪里。
有可能在 modules/下，也可能在 sites/all/modules/下面。这时候就需要用它来 帮你找出模组的路径，好引入它资料夹里面的档案。
如：

```
$my_module_path = druapl_get_path('module', 'my_module');
$my_theme_path = druapl_get_path('theme', 'my_theme');
$phptemplate_engine_path = druapl_get_path('theme_engine', 'phptemplate');
```

附带一提的，如果使用了「简洁网址」，就是网址不出现?q=xxx 的模式。连结使用相对路径的时候，可能会出现找不到档案的情形。(浏览器被网址列的网址搞混了)
这个时候建议搭配 base_path()这个函数，写一个相对于网页伺服器的"绝对路径"。
如：

```
$my_module_path = base_path() . drupal_get_path("module", "my_module");
//如果网站是在 drupal 资料夹下， base_path()会传回"/drupal"
```

路径用"/"开头，就表示由网页伺服器的"根"开始算相对路径。
如： "/drupal/my_module/test_page/"

不过如果使用 drupal_add_js() 函数，因为它本身就会加上网站本身的路径，所以就不需要加上 base_path()了。

程式码

```
<?php
function drupal_get_path ( $type , $name ) {
  return dirname ( drupal_get_filename ( $type , $name ));
}
?>
```

drupal_set_message

定义

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

includes/bootstrap.inc

描述

定义一组讯息，以反映刚执行命令的状态

参数

$message:要显示的讯息

$type:类别，可以是

- 'status'
- 'error'

传回值

所有已经定义的讯息

范例

例如要告诉使用者”你已经成功注册了”:

```
<?php
drupal_set_message ( t ( 'Created a new user account. No e-mail has been sent.' ));
?>
```

一个绿色的方框就会出现在下一个用户的可视页面内 开发者不用操心显示部份

又例如要显示一个错误：

```
<?php
drupal_set_message ( t ( 'Invalid password.' ) , 'error' );
?>
```

后面的一个参数如果为'error'
一个错误的红色框就会出现

注：第一个参数可以为 html

注：t()函数指可翻译的

程式码

```
<?php
function drupal_set_message ( $message = NULL , $type = 'status' ) {
  if ( $message ) {
    if (!isset( $_SESSION [ 'messages' ])) {
      $_SESSION [ 'messages' ] = array( );
    }     if (!isset(

$_SESSION [ 'messages' ][ $type ])) {
      $_SESSION [ 'messages' ][ $type ] = array();
    }

    $_SESSION [ 'messages' ][ $type ] [] = $message ;
  }

  //如果发生资料库连接错误，则讯息不会被记录
  return isset( $_SESSION [ 'messages' ]) ? $_SESSION [ 'messages' ] : NULL ;
}
?>
```

附加档案	大小
drupal_set_message_1.JPG	6.59 KB

drupal_set_title

定义

drupal_set_title($title = NULL)

includes/path.inc,第 187 行之后

描述

设定页面的标题文字(不是网站标题)

参数

$title 选择性参数，如果为 NULL 则不会改变概有的标题内容。

传回值

没有传回值，但会改变当下页面的标题。

范例

drupal_set_title("新标题");

当同一个页面要有动态标题内容时…
如：大家都看到都是同样"我的帐号"连结，但是开启后的页面标题变成自己的帐号名称。
就需要例用 drupal_set_title 去修改页面的标题，不然预设标题的文字和连结名称是一样的。

原始码

```
<?php
function drupal_set_title ( $title = NULL ) {
  static $stored_title ;   if (isset(

$title )) {
    $stored_title = $title ;
  }
  return $stored_title ;
}
?>
```

module_foo

看名字就知道这些函数是用来处理和模组有关的事情。
如果今天开发的模组需要和其它的模组互动，或是"相依"在其它的模组之上，或是要触发其它模组的

hook 。
那么这个部分的函数就要花时间去研究。

module_exists

定义

module_exists($module)

includes/module.inc

描述

判断某个模组是否存在(并已启用)

参数

$module: 模组的名字(不要带上.module 的副档名).

传回值

如果该名称的模组已安装且正被启用，则传回 TRUE 。

程式码

```
<?php
function module_exists ( $module ) {
  $list = module_list ();
  return array_key_exists ( $module , $list );
}
?>
```

theme user login block form

之前在怎样 styling user login block ?讨论内回应的内容上有一些错误
回应的内容已经没办法编辑
所以修改后另外 PO 一篇

在版型资料夹下的 template.php 内加上

```
function theme_user_login_block($form){
//自订登入画面 block
return _phptemplate_callback('login_block_form', array('form' => $form));
}
```

然后在版型资料夹下增加一个 login_block_form.tpl.php 的档案作为样板档
直接在样板档里排成我想要的样子

```
<div id="custom-login-bar">
<div class="form-label">帐号</div> <?php print drupal_render ( $form [ 'name' ]) ?>
<div class="form- label">密码</div> <?php print drupal_render ( $form [ 'pass' ]) ?>
<div class="form-item"> <?php print drupal_render ( $form [ 'submit' ]) ?> <?
php print drupal_render ( $form [ 'form_id' ]) ?> </div> </div>
```

drupal 的表单必须把$form['form_id'] & $form['form_token']必须 print 出来表单才有作用
不过 user_login_block 似乎是例外表单结构内原本就没有$form['form_token']
所以这边只 render $form['form_id']

另外搭配 imagebutton 还可以把原本的 submit 按钮改成图片

```
<?php
  $form [ 'submit' ][ '#type' ] = 'imagebutton' ;
  $form [ 'submit' ][ '#image' ] = 'xxx/xxx.jpg' ;
  print drupal_render ( $form [ ' submit' ]);
?>
```

注册新帐号&忘记密码的连结也都是定义在$form 里面
在样板档里 print_r($form)看一下就知道了

然后再加入 css (这边只是简单示意一下详细的 css 请自己再作设定)

```
<style>
#custom-login-bar .form-item label {
  display:none;
} #custom-login-bar .form-item, #custom-login-bar .form-label {   float:left; } </style >
```

要直接加在样板档的档案内
或版型的 style.css 档案内都可以

如果是只有一个样板档用到的 style
个人是喜欢直接放在样板档里

在样板档里就像一般的网页制作一样
随你怎么排列
只要知道 element 的名字
想在哪 render 都可以也不用照$form 里的顺序
可以改出单用 css 或 hook_form_alter 没办法达成的排版

虽然在$form['name']['title']可以把标题改成空字串而不显示标题
但是不建议这样做
因为表单的错误讯息会没有栏位的名称
例如请填写密码栏位的错误讯息会变成请填写栏位
如果表单的栏位有 5 个栏位没填就会变成

请填写栏位
请填写栏位
请填写栏位
请填写栏位
请填写栏位

根本没办法辨识是哪个栏位未输入
所以#title 还是留着比较好
不想显示标题的时候建议还是用 css 把 label 隐 藏掉

user_foo

user 也是模组之一，管理使用者的登入登出，或是权限的认证。
总之想作好一个有管理使用者的模组，这个部分的函数就必须有所了解。
这些函数不需要特别去引用，直接呼叫函数的名称就可以了。

user_access

定义

user_access($string, $account = NULL)
modules/user/user.module,第 351 行开始

描述

判断使用者是否具有某种权限。
所有的权限判断，都应该要例用这个函数。一来是让所有的程式有一致性，而且可以保证 SuperUser 能拥有所有的权限。

参数

$string:要判定的权限名字，如"administer nodes"
$account (预设):一个 User 物件。在要判断的使用者并不是「目前登入的那个人」时使用。

传回值

如果判定是「有这个权限」，则传回布林值 TRUE。

范例

检查目前使用者是否有「存取管理页面(administer comments)」的权限

$has_permission = user_access("administer comments"); // 第二个参数省略

原始码

```
<?php
function user_access ( $string , $account = NULL ) {
  global $user ;
  static $perm = array();   if (

is_null ( $account )) {
    $account = $user ;
  }

  //超级使用者 SuperUser 拥有所有的权限
  if ( $account -> uid == 1 ) {
    return TRUE ;
  }

  //为了灭少到资料库作查询连线数，会把作查过的资料放入静态变数中
  if (!isset( $perm [ $account -> uid ])) {
```

```
    $result = db_query ( "SELECT DISTINCT(p.perm) FROM {role} r INNER JOIN {permission} p
ON p.rid = r.rid WHERE r.rid IN ( %s)" , implode ( ',' , array_keys ( $account -> roles )));

    $perm [ $account -> uid ] = '' ;
    while ( $row = db_fetch_object ( $result )) {
      $perm [ $account -> uid ] .= "$row->perm, " ;
    }
  }   if (isset(

$perm [ $account -> uid ])) {
    return strpos ( $perm [ $account -> uid ], "$ string, " ) !== FALSE ;
  }   return

FALSE ;
}
?>
```

user_load

定义

user_load($array = array())

modules/user/user.module

描述

取得一个 user 物件

参数

$array 一个关连式阵列，让这个函数可以找得到你想找的使用者资料，可能是帐号名称或是 e-mail 位址。

传回值

如果找到的话，传回一个"完整"的 user 物件，否则传回 FALSE

范例

其实 Drupal 的登入就是靠它来检查，输入帐号密码，来作搜寻有没有符合的条件。

$account = user_load(array('name' => '纳格髓', 'pass' => 'very secret', 'status' => 1))

name：帐号
pass：密码
status：状态(1 代表启用中)

如果有找到，就表示帐号密码正确，$account 就会传回一个 user 物件。再下两行

global $user;
$user = $account; //把找到的 user 物件"存"起来

这样就算是完成登入了。很简单吧!!

附带一提，登出的语法也很简单。

global $user;
$user = drupal_anonymous_user();

程式码

```
<?php
function user_load ( $array = array()) {
  // Dynamically compose a SQL query:
  $query = array();
  $params = array();   foreach (

$array as $key => $value ) {
    if ( $key == 'uid' || $key == 'status' ) {
      $query [] = "$key = %d" ;
      $params [] = $value ;
    }
    else if ( $key == 'pass ' ) {
      $query [] = "pass = '%s'" ;
      $params [] = md5 ( $value );
    }
    else {
      $query []= "LOWER($key) = LOWER('%s') " ;
```

```
      $params [] = $value ;
    }
  }
  $result = db_query ( 'SELECT * FROM {users} u WHERE ' . implode ( ' AND
' , $query ), $params );   if (

db_num_rows ( $result )) {
    $user = db_fetch_object ( $result );
    $user = drupal_unpack ( $user );

    $user -> roles = array();
    if ( $user -> uid ) {
      $user -> roles [ DRUPAL_AUTHENTICATED_RID ] = 'authenticated user ' ;
    }
    else {
      $user -> roles [ DRUPAL_ANONYMOUS_RID ] = 'anonymous user' ;
    }
    $result = db_query ( 'SELECT r.rid, r.name FROM {role} r INNER JOIN {users_roles} ur ON
ur.rid = r.rid WHERE ur.uid = %d' , $user -> uid );
    while ( $role = db_fetch_object ( $result )) {
      $user -> roles [ $role -> rid ] = $role -> name ;
    }
    user_module_invoke ( 'load' , $array , $user );
  }
  else {
    $user = FALSE ;
  }   return

$user ;
}
?>
```

user_roles

定义

user_roles($membersonly = 0, $permission = 0)

modules/user/user.module,第 1762 行开始

描述

传回符合条件的「群组(身分)」阵列

(在 Drupal 里，群组(身分)被叫作"role")

参数

$membersonly:如果设为 TRUE ，则传回的时候会排除「访客群组(身分)」

$permission:一个字串，如果加到它，则只传回包含这个权限的群组(身分)回来

传回值

一个关连阵列，键是该群组在资料库的 ID，值是它的名字。

范例

列出现在所有的群组： (假设我事前建立一个叫作 power user 的群组)

```
print_r(user_roles());
```

传回值：

```
Array
(
    [1] => anonymous user
    [2] => authenticated user
    [3] => power user
)
```

程式码

```
<?php
function user_roles ( $membersonly = 0 , $permission = 0 ) {
  $roles = array();   if (

$permission ) {
    $result = db_query ( "SELECT r.* FROM {role} r INNER JOIN {permission} p ON r.rid =
p.rid WHERE p.perm LIKE '%%%s%%' ORDER BY r.name" , $permission );
  }
  else {
    $result = db_query ( 'SELECT * FROM {role} ORDER BY name' );
  }
  while ( $role = db_fetch_object ( $result )) {
    if (! $membersonly || ( $membersonly && $role -> rid != DRUPAL_ANONYMOUS_RID )) {
      $roles [ $role -> rid ] = $ role -> name ;
    }
  }
  return $roles ;
}
?>
```

user_save

定义

user_save($account, $array = array(), $category = 'account')
modules/user/user.module,第 106 行开始

描述

更新一个使用者帐号内容，或是建立一个新的帐号

参数

$account 一个 user 物件，如果$user->uid(帐号 ID)为空，则建立新的帐号，如果有东西的话，就更新这个 uid 的帐号资料

$array 一个带有帐号资讯的阵列。例如： array('name' => 'My name');值为 NULL 的话，就表示把这个栏位清空。
(注：在更新中，没有列在这个阵列里的栏位会保持不变)
$category (选择性参数)用来作注记的参数。在这个函数中并没有用到，主要是用来当参数传给相关的 hook 函数。

范例

第一个参数要求的是一个 user 物件，但是其实只用到里面的 uid 属性。
所以要新增一个帐号的时候，并不用费心去建立一个"空的 user 物件"，给它一个'' (空字串)就好了。
如：

```
$roles = array(
   '3' => '角色名称',
   '4' => '角色名称',
   '5' => '角色名称',
);
user_save('', array("name" => '纳格髓', "pass" => 'unknow', "status" => 1, 'roles' =>
$roles , 'profile_tel' => '0800 123456'));;
```

name：帐号名称
pass：密码(不需要作加密动作)
status：状态(1 表示启用中)
roles：权限，也许叫它"群组"比较适合，这里要放 array。(例子中，是加入编号 3, 4, 5 的群组)
profile_tel：这个是自订的使用者栏位，名字叫作"tel"

注：
在 Drupal 的群组(roles)中，1 表示访客，2 表示已注册帐号。这两个资讯在记录的时候会"自动跳过"。
因为有没有帐号就已经足以作为这两者的区分。所以就算你在 roles 放入 1 或 2 也不会被记入资料库中。

(原来文章的例子是直接打 array(3, 4, 5)来加入编号 3, 4, 5 的角色，这种写法是错误的)

原始码

```
<?php
function user_save ( $account , $array = array(), $category = 'account' ) {
  // Dynamically compose a SQL query:
  $user_fields = user_fields ();
  if ( $account -> uid ) {
```

```
    user_module_invoke ( 'update' , $array , $account , $category );

    $data = unserialize ( db_result ( db_query ( 'SELECT data FROM {users} WHERE uid =
%d' , $account -> uid )));
    foreach ( $array as $key => $value ) {
      if ( $key == 'pass' && !empty( $value )) {
        $query .= "$key = '%s', " ;
        $v [] = md5 ( $ value );
      }
      else if (( substr ( $key , 0 , 4 ) !== 'auth' ) && ( $key != 'pass' )) {
        if ( in_array ( $key , $user_fields )) {
          // Save standard fields
          $query .= "$key = '%s', " ;
          $v [] = $value ;
        }
        else if ( $key != 'roles' ) {
          // Roles is a special case: it used below .
          if ( $value === NULL ) {
            unset( $data [ $key ]);
          }
          else {
            $data [ $key ] = $value ;
          }
        }
      }
    }
    $query .= "data = '%s' " ;
    $v [] = serialize ( $data );

    db_query ( "UPDATE {users} SET $query WHERE uid = %d" , array_merge ( $v ,
array( $account -> uid )));

    // Reload user roles if provided
    if ( is_array ( $array [ 'roles' ])) {
      db_query ( 'DELETE FROM {users_roles} WHERE uid = %d' , $account -> uid );
foreach (
```

```
array_keys ( $array [ 'roles' ]) as $rid ) {
        if (! in_array ( $rid ,
array( DRUPAL_ANONYMOUS_RID , DRUPAL_AUTHENTICATED_RID ))) {
          db_query ( 'INSERT INTO {users_roles} (uid, rid) VALUES (%d, %d)' , $account -
> uid , $rid );
        }
      }
    }

    // Delete a blocked user's sessions to kick them if they are online.
    if (isset( $array [ 'status' ]) && $array [ 'status' ] == 0 ) {
      sess_destroy_uid ( $account -> uid );
    }

    // If the password changed, delete all open sessions and recreate
    // the current one.
    if (isset( $array [ 'pass' ])) {
      sess_destroy_uid ( $account -> uid );
      sess_regenerate ();
    }

    // Refresh user object
    $user = user_load (array( 'uid' => $account -> uid ));
    user_module_invoke ( 'after_update' , $array , $user , $category );
  }
  else {
    $ array [ 'uid' ] = db_next_id ( '{users}_uid' );     if (!isset(

$array [ 'created' ])) {    // Allow 'created' to be set by hook_auth
      $array [ 'created' ] = time ();
    }

    // Note, we wait with saving the data column to prevent module-handled
    // fields from being saved there. We cannot invoke hook_user('insert') here
    // because we don't have a fully initialized user object yet.
    foreach ( $array as $key => $value ) {
      switch ( $key ) {
```

```
        case 'pass' :
          $fields [] = $key ;
          $values [] = md5 ( $value );
          $s [] = "'%s'" ;
          break;
        case 'uid' : case 'mode' : case 'sort' :
        case 'threshold' : case 'created' : case 'access' :
        case 'login' : case 'status ' :
          $fields [] = $key ;
          $values [] = $value ;
          $s [] = "%d" ;
          break;
        default:
          if ( substr ( $key , 0 , 4 ) !== 'auth' && in_array ( $key , $user_fields )) {
            $fields [] = $key ;
            $values [] = $value ;
            $s [] = "'%s'" ;
          }
          break;
      }
    }
    db_query ( 'INSERT INTO {users } (' . implode ( ', ' , $fields ) . ') VALUES (' . implode ( ',
' , $s ) . ')' , $values );

    // Build the initial user object.
    $user = user_load (array( 'uid' => $array [ 'uid' ]));

    user_module_invoke ( 'insert' , $array , $user , $category );

    // Build and save the serialized data field now
    $data = array() ;
    foreach ( $array as $key => $value ) {
      if (( substr ( $key , 0 , 4 ) !== 'auth' ) && ( $key != 'roles' ) &&
(! in_array ( $key , $user_fields )) && ( $value !== NULL )) {
        $data [ $key ] = $value ;
      }
    }
```

```
    db_query ( "UPDATE {users} SET data = '%s' WHERE uid =
%d" , serialize ( $data ), $user -> uid );

    // Save user roles (delete just to be safe).
    if ( is_array ( $array [ 'roles' ])) {
      db_query ( 'DELETE FROM {users_roles} WHERE uid = %d' , $array [ 'uid' ]);
      foreach ( array_keys ( $array [ 'roles' ]) as $rid ) {
        if (! in_array ( $rid ,
array( DRUPAL_ANONYMOUS_RID , DRUPAL_AUTHENTICATED_RID ))) {
          db_query ( ' INSERT INTO {users_roles} (uid, rid) VALUES (%d,
%d)' , $array [ 'uid' ], $rid );
        }
      }
    }

    // Build the finished user object.
    $user = user_load (array( 'uid' => $array [ 'uid' ]));
  }

  // Save distributed authentication mappings
  $authmaps = array();
  foreach ( $array as $key => $value ) {
    if ( substr ( $key , 0 , 4 ) == 'auth' ) {
      $authmaps [ $key ] = $value ;
    }
  }
  if ( sizeof ( $authmaps ) > 0 ) {
    user_set_authmaps ( $user , $authmaps );
  }   return

$user ;
}
?>
```

常用的全域变数

在写模组的同时，发现有些资讯是记在全域变数里面，在遍寻的函数找不到之后才发现。
所以把一些常用的全域变数整理到这里，提供给人查询。
所有用变数使用之前，必须要先「global $变数;」才能使用...

$user

使用者物件
既使没有人登入，也会有一个"访客"况态的物件。要知道这个人有那些资讯，就必须要使用这个物件。

$base_url

网站根目录网址
当使用「clean_url(简絜网址)」的时候，有时候会需要用到绝对路径来作网页连结。
不确定网址的开头是什么的时候，像是不确定自己的模组会被装到哪个站里面，可以用它来 查询网站根目录的网址。
如果我用把 drupal 装在"drupal_test"的目录下，在本机测试时，这个变数可能会是「http://localhost/drupal_test」。

$base_path

网站根目录网址(相对路径)
当使用「clean_url(简絜网址)」的时候，有时候会需要用到绝对路径来作网页连结。
不确定网址的开头是什么的时候，像是不确定自己的模组会被装到哪个站里面，可以用它来 查询网站根目录的网址。
如果我用把 drupal 装在"drupal_test"的目录下，在本机测试时，这个变数会是「drupal_test」。

慢慢增加中....

把玩 Drupal 模组： Hook System 运作简介

模组系统是 drupal 很重要的运作方式，drupal 依靠着少少的核心程式，便能让模组能做到任何事情。

drupal 只有 21 个档案在 include 里头，每次必会 loading 进来，其他的全都放在 modules。
也就是说，除了那几只档案以外，全部的东西都把他当成 module 在写。诸如 CMS 最基本的功能，文章管理、评论、讨论区、分类...等的功能，全部都写在 module 里，include 里头所提供的是各种 api，档案处理函式、资料库存取、表单生成...等等，这样的分层，module 便可以专心的开发各种功能。

当然，这样的架构不够令人注目。有许多 web app 架构，对于模组（module）、插件（plug-in）...等的运作，通常是让他们各自为政，自己干自己的事情。多是用核心提供的 object 和 function，加上 module 自己额外的 code，达到 module 要做到的额外功能。但是 drupal 的核心运作却不是如此。

drupal 处理使用的程式为 modules/user.module。如果今天想要在看使用者资料的同时，也想看看所有使用者过去发表文章的 list，那该怎么写呢？

直接一点，更改 user.modue，在显示时，顺便去文章资料库抓相关的资料？然而这样却不是一个好方式，今天任何想要对使用者增加新功能的时候，都得 trace 一次 user.module 的 code，看懂他在干啥，然后把新的 code 安插在合适的地方.. .最后可能增加 user.module 的复杂度，增加维护那支 module 的难度，共同开发时，更是一个危险的方式。

第二种方式，重写一个新的浏览页面，重新写一个 SELECT 的语句，让 SELECT 的时候除了使用者资讯，也把文章资料一起抓出来，然后显示到不同的页面。但是这样很浪费，明明跟 user.module 重复的功能达到一半以上，那是不是之后要新增功能，都得重写一次呢？

上面两种方式在 drupal 中也都可以达成，然而熟悉 Drupal 的人却不会如此。Drupal 的开发者很聪明，他的模组系统（module system）考虑到了模组再利用这一点，每个模组都视为可以再利用的资源，只要写 module 的人想写，透过模组系统便可以跟所有的 module 交互作用。

第三种方式以 modules/user.module 为例，他即是处理包含新增、修改、删除、注册、登入....等所有与使用者相关的功能。在进行每个重要的功能时，user.module 都会呼叫一个函式去扫所有的 module，看看是否有其他的 module 要在 user.module 进行此动作时，也进行一些其他想要做的事情，这就是 drupal 重要的 Hook System。example: 在 drupal user.module 里头可以找到如下的程式码

```
function user_view($uid = 0) {
  // ... skip
  // moudle_invoke 扫描所有的 module
  // 看看有没有 modulename_user 这个 function
  // 有个话就看'view'这个功能的部份要加上什么
  foreach (module_list() as $module) {
    if ($data = module_invoke($module, 'user', 'view', '', $account)) {
      // do something...
    }
  }
  // ... skip
}
```

所以，第三种方式，不用重写，也不用改到 user.module，只要自己新增 module 和写一个 function，便可以轻松让浏览使用者资讯时，加上过往文章。example: 新增自己的 module，与 hook system 紧密运作 新增 sample.module

```
function sample_user($type, &$edit, &$user, $category = NULL) {
```

```
  if ($type == 'view') {
    return /*过往文章，型别为一阵列*/;
  }
}
```

这就是 drupal 把众多主要功能都写成 module 的原因，让所有模组之间都可以交互利用，或是写给别人利用，或是利用别人的 module，像积木一样推砌成想要的功能，却又不浪费资源。

参考资源：
详细的用法在：http://drupaldocs.org/api/head/function/hook_user Module developer's guide：http://drupal.org/node/508

不使用 views，如何自订节点清单显示页面

原文连结：http://www.500959.com/node/510
基本上，views 和 cck 是大家都建议用的两个 drupal 模组，因为这两个模组太强大了，一个可以自由定制栏位，一个可以自由过滤资料用以显示， drupal 的许多其它模组都是基于这两个的。正因其强大，所以也庞大，庞大耗费资源，相对来说，配置起来也较复杂。对于一些把 drupal 做为个人博客来用的朋友来说，通常不想使用这两个模组，但 drupal 预设就只有一种排列文章的方式，按时间发表顺序，而 bloger 们可能就需要有多一些的排列显示方式。

这个时候可以用查询资料库再配合 drupal 的一些核心函数来达到简单过滤显示的目的。举个例子吧，用习惯了国内 cms 的朋友，都喜欢在首页上显示一些区块，什么最新文章、最新推荐、最新评论、热门文章……

咱们先创建一个 page 节点，标题就看你的爱好了，随便取吧。内容呢，就随便写段代码吧：

```
<?php
echo '嘿罗，世界'; //反正一般程式测试都这个套路。
?>
```

然后在输入格式里选择 php code 模式，自订路径呢，随便想一个吧，这儿就定义为:index.html。提交保存，这不就一个“嘿罗世界”吗？别着急，这只是第一步。

现在进入第二步，到“网站资讯（admin/settings/site-information）”中，拉到最后面，把默认首页设置为咱们刚才创建的页面路径：index.html，好，现在打开网站，发现默认首页就剩“嘿罗世界”了，忽悠人啊这不是。别急，接下来做第三步。

第三步，开始往里边添加内容了。我想添加个最新 blog 文章的列表。drupal 区块里有个默认的最新 blog 文章，到区块中，查看最新 blog 文章的区块连结是这样的：
admin/build/block/configure/blog/0，注意最后两层（blog/0） ，这很重要。这表示这个区块是

由 blog.module 生成的第 0 个区块（从 0 开始计数的）。现在我们编辑那篇文章，把里边的“嘿罗世界”可以删除了，放这段代码进去：

```
<?php
$block = module_invoke ( 'blog' , 'block' , 'view' , 0 );
echo $block [ 'subject' ]; //显示区块的标题
echo $block [ 'content' ]; //显示区块的主内容区。
?>
```

纯粹的引用代码让人有些糊涂，解释一下，module_invoke：载入模组，blog：点名要载入这个；block：区块，view：显示。后面的就不难理解了，要显示这个模组的第 0 个区块。就这么简单，要直接在文章中插入其它区块也是这个方法。现在提交保存，到首页看看，是不是出现了最新 blog 文章的列表（当然，你得先发表几篇 blog 文章）。

我想要显示其它类型，比如 story 的最新文章呢？因为默认没有提供区块，所以一般的做法呢是用 views 来过滤出来，但文章一开头就说了，咱们不用 views。这就进入另一个重点部分，读取资料库来显示。编辑文章，在后面插入这段代码：

```
<?php
echo '<h2>最新 story</h2>' ; //标题随意
$result = db_query_range ( "SELECT n.nid, n.title FROM {node} n WHERE n.type = 'story'
ORDER BY n .created DESC" , 10 );
while ( $test = db_fetch_object ( $result )) {
    echo l ( $test -> title , 'node/' . $test -> nid ). '<br>' ;
    };
? >
```

drupal 查询资料库语句重新定义过，但基本上和 mysql 手册上的方法差不多。完整的手册请看这儿。现在说我们上面的例子，文章的基本资讯都存储在 node 这个表里，因为我们只要显示标题清单，所以查询 title 和 nid 栏位，条件呢就是限制为 story 类型。你可以把条件改为特定的用户(uid)，或是否推荐(promote)、置顶(sticky)，或是否有评论(comment)等等方式。最后是排序，这个例子中我们按节点的创建时间倒序，也就是最新文章在前面，数目限制为 10 条。下面就是一个阵列回圈了，在这里边可以自行排版，添加 css 等。

保存，现在看看效果，最新 story 文章列表是不是出现了呢？而热门内容呢，statistics 模组也提供了一个按点击排序的文章列表，我们可以直接用上面插入区块的办法把它放到页面里。现在我们页面里有最新文章、最新推荐、最新日志、热门内容、最新评论等区块列表了，看起来像那么回事了，接下来的事就是排版了。这个就取决于个人的审美观了，熟悉 css 的就用 css，不熟悉的就直接用表格套上去就行了。

可以收工了，可我还想有个 more，你看大多数网站的列表下面，不都有个 more，点击进去，显示更多的内容，并且这里边的内容还是可以分页的。好，现在我们来解决这个问题。还是举例子吧，也想不到其它更好的手段了。先在首页的清单下面加个 more 连结：

```
<?php
echo '<h2>最新 story</h2>' ;
$result = db_query_range ( "SELECT n.nid, n.title FROM {node} n WHERE n.type = 'story'
ORDER BY n.created DESC" , 10 );
while ( $test = db_fetch_object ( $result )) {
    echo l ( $test -> title , 'node/' . $test -> nid ). '<br>' ;
    };
echo '<a href ="/story/all">更多</a>' ;
?>
```

就这么简单？是的，就这么简单，哈哈。可怎么也得加个判断吧，万一只有一篇文章，或者只有 0 篇文章呢，也显示个“更多”，那多傻。那就修改一下：

```
<?php
echo '<h2>最新 story</h2>' ;
$result = db_query_range ( "SELECT n.nid, n.title FROM {node} n WHERE n.type = 'story'
ORDER BY n.created DESC" , 10 );
if ( db_num_rows ( $result ) > 0 ) { //加入判断，查询结果大于 0 才显示；
while ( $test = db_fetch_object ( $result )) {
    echo l ( $test -> title , ' node/' . $test -> nid ). '<br>' ;
    };
}else { //否则就显示
echo '没有文章' ;
} if(

db_num_rows ( $result ) > 10 ){ //如果大于 10 篇，就显示更多连结。
    echo '<a href="/story/all">更多</a>' ;
}
?>
```

提交保存，有了个“更多”连结，可点选连结不是“找不到页面”吗？别急，下一步。创建一个 page 节点，标题就写个“新闻列表”吧，输入格式还是选取为 php code，自订路径呢，就要和上面的一致了：story/all。现在要填内容进来了，怎么把这个最新的 story 文章清单显示出来呢？

```
$result = pager_query ( "SELECT n.nid, n.title FROM {node} n WHERE n.type = 'story'
ORDER BY n.created DESC" , 15 );
if ( db_num_rows ( $result ) > 0 ) { //加入判断，查询结果大于 0 才显示；
while ( $test = db_fetch_object ( $result )) {
    echo l ( $test -> title , ' node/' . $test -> nid ). '<br>' ;
    };
}else { //否则就显示
echo '没有文章' ;
}
echo '<p>' . theme ( 'pager' , NULL , 15 ). '</p>' ;
?>
```

请注意这一段代码，和上面其它的对比，有一些变化，首先是 db_query_range 变成了 pager_query，对于有分页需求的，都使用这个函数查询。其次当然是下面多了翻页函数：theme('pager', NULL, 15)。在查询语句中我们定义了数目是 15 条，翻页里当然也是设置 15 条。现在保存这个节点，看看，是不是出现了列表，并且有翻页了（当然，前提是你的 story 文章得多于 15 条，如果不够多，把 15 改成 2 或 3 试试）。

这个时候可能又会觉得一个清单就显示个标题，未免太单调了，我还想显示点作者啊，发表时间啊，评论数目啊，点击数量啊等等。好吧，咱们来完成这个需求。

```
<?php
echo '<h2>新闻列表</h2>' ;
$result = pager_query ( "SELECT n.nid, n.title,n.comment, n.created,u.uid,
u.name,s.totalcount FROM {node} n INNER JOIN {node_counter} s ON n.nid = s.nid INNER
JOIN {users} u ON n.uid = u.uid WHERE n.type = 'story' ORDER BY n.created DESC" , 15 );
if ( db_num_rows ( $result ) > 0 ) { //加入判断，查询结果大于 0 才显示；
while ( $test = db_fetch_object ( $result )) {
echo '标题：' . l ( $test -> title , 'node/' . $test -> nid ).
        '  作者：' . l ( $test -> name , 'user/' . $test -> uid ).
        '  评论：  ' . $test -> comment .
        '  点击：' . $test -> totalcount .
        '  发表时间：' . format_date ( $test -> created ). '<br>' ;
    };
}else { //否则就显示
echo '没有文章' ;
}
```

```
echo '<p>' . theme ( 'pager' , NULL , 15 ). '</p>' ;
?>
```

细心的你肯定发现了，这里使用了多表查询，因为节点的点击量是存放在另外一个表里的，而作者的资讯又存在 users 表里。老套路，提交保存，现在看看列表，是不是多了作者、评论这些资讯。只是排版未免太难看了。那就用 css 自己调整吧，可我又不想使用 css，而且这种清单式的显示，使用表格更有优势，方便又快捷。那就用列表吧。这样改一改：

```
<?php
echo '<h2>新闻列表</h2>' ;

$header = array( //这里增加了，先定义个表格头部
    array( 'data' => '标题' ),
    array( 'data' => '作者' ),
    array( 'data' => '评论' ),
    array( 'data' => '点击' ),
    array( 'data' => '发表时间' )
  );
$tablesort = tablesort_sql ( $header );
$result = pager_query ( "SELECT n.nid, n.title,n.comment, n.created,u.uid,
u.name,s.totalcount FROM {node} n INNER JOIN {node_counter} s ON n.nid = s.nid INNER
JOIN {users} u ON n.uid = u.uid WHERE n.type = 'story' ORDER BY n.created
DESC" . $tablesort , 15 );
if ( db_num_rows ( $result ) > 0 ) { //加入判断，查询结果大于 0 才显示；
while ( $test = db_fetch_object ( $result )) {
   //这儿不直接列印，而是定义成一个阵列了。
$rows [] = array( 'data' =>
          array(
             l ( $test -> title , 'node/' . $test -> nid ),
             l ( $test -> name , 'user/' . $test - > uid ),
             $test -> comment ,
             $test -> totalcount ,
             format_date ( $test -> created ),
          ),
       );
   };
}else { //否则就显示
echo '没有文章' ;
```

```
}
echo theme ( ' table' , $header , $rows ); //这儿列印出表格。
echo '<p>' . theme ( 'pager' , NULL , 15 ). '</p>' ;
?>
```

提交保存，现在看一看页面，是不是都在一个表格里，排版都省了，整整齐齐。这样就差不多了吧，又有了区块，又有了列表页。不过，也许你突然又觉得全是标题列表有也点单调，还想看看摘要显示是什么效果。咱们就来把标题清单改为摘要模式，这个更简单一点，直接使用 node_view 和 node_load，这两个函数，只要你告诉它节点 nid，它就能载入节点的摘要或全文视图了。省得麻烦，就直接还是用这个新闻列表页来做试验吧。编辑节点，放入这段代码：

```
<?php
echo '<h2>新闻列表</h2>' ;
$result = pager_query ( "SELECT n.nid FROM {node} n INNER JOIN {node_counter} s ON
n.nid = s.nid INNER JOIN {users} u ON n.uid = u.uid WHERE n.type = 'story' ORDER BY
n.created DESC" , 15 );
if ( db_num_rows ( $result ) > 0 ) { //加入判断，查询结果大于 0 才显示；
    while ( $test = db_fetch_object ( $result )) {
        $output .= node_view ( node_load (array( 'nid' => $test -> nid )), 1 ); //这儿的 1 或 0 是
全文或摘要。
   };
}else { //否则就显示
    echo '没有文章' ;
}
echo '<p>' . theme ( 'pager' , NULL , 15 ). '</p>' ;
?>
```

注意几个变化，首先要查询资料库的时候，只需要节点 nid 就行了。其次是在阵列回圈时，直接用 node_view 和 node_load 来显示出文章。这儿就用不着排版了，全部是按照你在 node.tpl.php 中定义的样式来显示。

现在终于可以结束了，好像在节点组织显示方面，也没其它的需求了。上面的内容我是一边试验一边写下来的，在 5.x 版本上，应该不会有错误。本来准备截图，嫌上传麻烦，就没截了。如果测试过程中有什么问题，请提出来。显示需求不是很多，可以使用这种查询资料库的方式，如果有许多自订的显示需求，建议还是用 views，毕竟它内置了缓存，而且和其它模组的互动也更好，当然，用起来也更方便。希望这篇文章对于喜欢 drupal，但又不想使用 views 的朋友有一定的帮助。

投票/评比模组的评比

Drupal 有很多种投票/评比模组，有的简单、有的复杂，各有巧妙不同与适用时机。该如何选择？

之前在做 nbalive.tw 时，花满多时间在测试拥有类似功能的不同模组，例如投票、聊天、相簿、讨论区，而这篇 Lullabot - A Review of Node Review Modules 令我有相见恨晚的感觉，因为他集合了 Drupal 的 6 个投票模组，做了简单的介绍与图示，方便使用者做初步判定。

简单介绍一下：Node Vote -简单的 1-10 分数投票。NodeReview -可设定多种评比项目（要是能画个五力分析图出来就更好了）。Simple Vote -更简单的五星级投票。userreview -类似 Amazon.com 的书评功能，投票并写心得，心得是一个独立的内容类型。Vote up/down Package -类似 digg.com，好或不好两种选择。Voting -也是五星级投票，可显示平均票数和我的投票。

Form 的美化心得

嗨!最近研究的是 form 的美化,延续上篇[问题],
解决之后真开心,分享一下我的美化的 code(用 CSS)
然后还是有几个问题:
(1)想弄更漂亮,想在 input 加上 onFocus 之类的,不知怎么加
(2)像我这样把 CSS 放在 tpl.php,会不会不好呢?

谢谢!

先看效果, 表单

这是按钮

下面是我的原始码:

template.php

```
<?php
function phptemplate_lifestyle_node_form ( $form ) {
  global $user ;
  $vars = array( 'user' => $user , 'form' => $form );
  return _phptemplate_callback ( 'lifestyle_form' , $vars );
}
?>
```

lifestyle_form.tpl.php

```
<?php
//drupal_set_message('<pre>' . print_r($form, true) . '</pre>');
drupal_set_title ( '新增宠物店家资料' );
?>
<style>
#content-area . information{
background:#F8FFEF;
width: 585px;
padding:10px;
margin:10px auto;
border:1px dotted #ccc;
}
#content-area .form-item{
width: 450px; /*width of right column containing the inputs */
clear: left;
margin: 0 auto;
padding: 5px 0 8px 0;
padding-left: 155px; /*width of left column containing the label elements*/
border-top: 1px dashed gray;
height: 1%;
}
#content-area label{
font-weight: bold;
float: left;
line-height:25px;
height:25px;
margin-left: -155px; /*width of left column*/
width: 150px; /*width of labels . Should be smaller than left column (155px) to create
some right margin*/
padding-left:10px;
}
#content-area input, #content-area textarea, #content-area select{
border:1px solid #ccc;
}
#content-area input:focus, #content-area textarea:focus{ /*for Firefox*/
```

```
background-color: #D5E7BD;
}
#content-area input{
width: 300px;
height:25px;
font-size:22px;
}
#content-area select{
height:25px;
font-size:17px;
}
#content-area textarea {
width: 440px;
}
#edit-body{ /*主要内文的高度预设好高哦！*/
height:100px;
}
#content-area .form-submit {
font-size: 12px;
background-color: #333333;
color: #FFFFFF;
width:100px;
height:30px;
position:relative;
left:250px;
top:20px;
}
</style>
<div class="information">
谢谢你提供店家资料哦！<br />
每次填写资料，我们都会怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样怎样，总之这个区块会放一些填表单的说明。
</div> <?php //把种类拉到最上面 ?> <?php print drupal_render ( $form [ 'taxonomy' ][ '4' ])
; //type ?> <?php print drupal_render ( $form [ 'taxonomy' ][ '3' ]); //location ?> <?php //
不要这些 ?> <?php drupal_render ( $form ['taxonomy' ]); //分类的外框，很丑? > <?php
drupal_render ( $form [ 'body_filter' ][ 'format' ]);  //内文说明，不要?> <?php //剩下的表单依
```

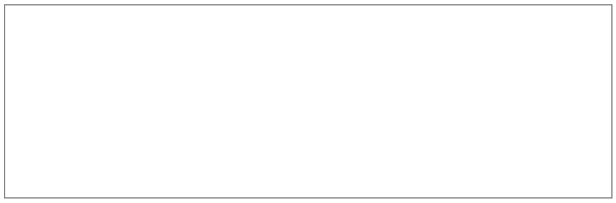
照权重排列 ?> <?php print drupal_render ($form); ?>

Drupal API 的 Dreamweaver 扩充套件

Hi!原文在这: Drupal API 的 Dreamweaver 扩充套件

今天发现了一个不错的东西，如果你是用 Dreamweaver 来 coding 的话，又是 Drupal 的使用者(不知这两个条件都成立的人多不多)，这玩意儿还满不错的噢！虽然帮不上大忙，不过，会在你输入一个函数的时候，闪一个小视窗告诉你这个函数里面要填的东西大概是什么。

像是这样。



这是这个 extension 的资讯。

> 名称：Drupal API extension for Dreamweaver
> 网站：http://xtnd.us
> 下载页面：这儿
> 直接下载：Drupal_API.mxp
> 官方说明：
> *在原始码模式时，档案类型是 Drupal 相关类型(.module, .php, . tal, .info, .inc, .theme, .js 等等)的话，就会出现提示。(是出现在打完函数和左边小括号之后)
> *您可在编辑>偏好设定>程式码提示那边停用这个功能。
> *您可以在说明>Drupal API for Dreamweaver 这个选单找到 Drupal API 网站连结。

安装方式应该不用讲了。依照这边的说法，似乎装了之后就会自动让 Dreamweaver 可以开 Drupal 的档案呢！所以我之前这篇「[Dreamweaver]让非 php 档名也能有 php 色彩标示」就白搭了。

以下是官方说明原文。

Drupal API Code Hints for Dreamweaver

Dreamweaver Versions: MX(6) - CS3(9)

This extension provides code hints for Drupal API's versions: 5.x, 6.x

* Drupal Code Hints appear in Code or Split view (Ctrl+Space) when editing Drupal files (.module, .php, .tal, .info, .inc, .theme, .js, etc.)

* You can deactivate Drupal versions that you aren't using in the Edit > Preferences > Code Hints menu .

* Help is available in the Help > Drupal API for Dreamweaver menu.

Updates and information about this extension available online at http://xtnd.us