

Drupal 模块开发-form api and hooks

-Drupal培训Lesson 5

Rapid Technologies

www.chinaldap.com

2011-5-13

Rapid Technologies



课程安排

- 表单系统 Form API
- 常用Hooks

本课程适合熟悉PHP语言语言而且熟悉模块开发基本概念用户

Form API

2011-5-13

Rapid Technologies



Drupal form 概括

Drupal的 form API提供了复杂的机制来提供如下的功能。

- 自定义表单的外观(**theme**)
- 表单的验证 (**validation**)
- 表单的执行
- 任何已经存在的表单都可以被改写，添加表单元素，重新排列顺序等。 (**alter**)

总之，drupal的form api 提供了一个安全的表单框架机制。

表单API - 生成表单

表单都是以数组（array）的形式来定义

```
function question_answer_edit_answer_form(&$form_state,  
    $q_nid='0') {  
    $form['answer'] = array(  
        '#type' => 'textarea',  
        '#title' => t('I have to answer'),  
        '#required' => true,  
        '#default_value' => "",  
    );  
    $form['q_nid'] = array(  
        '#type' => 'hidden',  
        '#value' => $q_nid,  
    );  
    $form['submit_answer'] = array(  
        '#type' => 'submit',  
        '#value' => t('Submit'),  
        '#weight' => 4, );  
    // $form['#theme'] = 'survey_form';  
    return $form;}
```



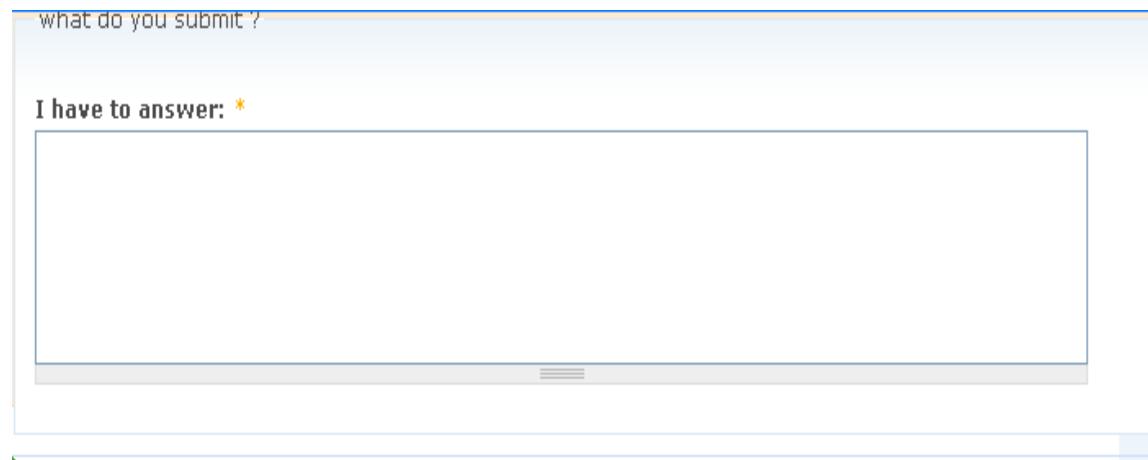
如果没有设置
#type, 则默认是markup

The screenshot shows a web page with a form. At the top, there is a text input field containing the placeholder "I have to answer: *". Below the input field is a large orange arrow pointing from the explanatory text on the left to the input field. At the bottom right of the form is a red "Submit" button.

表单API - 生成表单注意事项

如果使用fieldset,则构建有层次的 (array)

```
$form['data'] = array(  
    '#type' => 'fieldset',  
    '#title' => t('what do you submit  
?'),  
);  
  
$form[data]['answer'] = array(  
    '#type' => 'textarea',  
    '#title' => t('I have to  
answer'),  
    '#required' => true,  
    '#default_value' => '',  
);  
...
```



The screenshot shows a web page with a form. At the top, there is a label "what do you submit ?". Below it, there is another label "I have to answer: *". A large text area is present for input. The entire form structure is enclosed in a blue border.

表单API – 验证提交的数据

- 如果用户提交数据后，想验证某些数据字段的值，可以在**form validate**来处理。
- 这个函数的名字是`<form ID > _validate()`
- 比如在我们的例子里

```
function question_answer_edit_answer_form_validate($form, &$form_state) {  
  if ($form_state['values']['answer'] == "") {  
    form_set_error("", t('You must fill in an answer.'));  
  }  
}
```

- 用户提交的数据存在 `$form_state['values']` 里
- `Form_set_error`可以用户重新回到填写表单页面并打印出错信息

表单API – 提交数据后的处理

- 用户数据验证通过后，可以使用`<form ID> _submit()`函数处理数据
- 在我们的例子里就是如下格式

```
function question_answer_edit_answer_form_submit($form, &$form_state) {  
    $q_nid = $form_state['values']['q_nid'];  
    $answer = $form_state['values']['answer'];  
    $form_state['redirect'] = ('node/'.$q_nid);  
}
```

- 这个函数被调用，仅仅是你在`form`里设置了`submit`而且`validate`通过后才被执行
- 这个提交函数被处理后，如果想跳转，则可以用`$form_state['redirect']='url'`跳转，如果不设置这个，则默认跳转到表单页面。

表单API – 数据处理的流程

- `drupal_get_form()` 函数处理所有的表单呈现和数据处理，这表示你所创建的表单数组串在呈现的时候，别这个函数所处理，同时，你提交的数据也被这个函数处理。
- 所有，在我们想呈现的时候，就需要这么来使用

```
$output = drupal_get_form('question_answer_edit_answer_form', $nid=2);
```
- `$nid=2`是个额外的参数，你可以把这个参数传到表单里。

表单处理-更改任意的表单

- `Hook_form_alter(&$form, &$form_state, $form_id)`， 如果你想修改任意其他一个模块或者系统的表单，你需要使用这个。在我们的例子里，`question_answer_form_alter`
- 第一步，你需要获取你要修改表单的`form_id`，
 - ✓ 获取方法1，这样会打出多个`form_id`,找到你需要的那个。

```
function question_answer_form_alter (&$form, &$form_state, $form_id) {  
  drupal_set_message("Form ID is : " . $form_id);  
}  
✓ 获取方法2： 查看html 源代码  
✓ 获取方法3： 使用devel模块
```
- 第二步： 使用`switch ($form_id) case`来对不同的表单进行处理，比如添加一个新的表单`textfiled`等等。修改默认值。

如果是仅仅处理一个表单，则可以使用这个`hook_form_FORM_ID_alter()`

表单处理 – 系统变量设置

- Variable_set('变量名', '默认值'), 设置一个变量
- Variable_get('变量名', '默认值'), 获取一个变量的值

```
function simplerate_admin_settings_form() {
  $form = array();

  $form['rate_node'] = array(
    '#type' => 'textfield',
    '#title' => t('Rate Node ID'),
    '#default_value' => variable_get('rate_node', '1'),
    '#description' => t('You may firstly create a page , you will get a node id,
and then      put here. the rating will be shown on this node'),
  );
  return system_settings_form($form);
}
```

其他常用Hooks



Hook_block – 定义自己的区块(block)

```
■ <?php
 /**
 * Implementation of hook_block().
 * @param string $op one of "list", "view", "save" and "configure"
 * @param integer $delta code to identify the block
 * @param array $edit only for "save" operation
 */
function onthisdate_block($op = 'list', $delta = 0, $edit = array()) {

switch($op) {
    case 'list':
        return $blocks
    case 'view'
        Break;
    }
?>
```

Hook_block

```
function question_answer_block($op = 'list', $delta = 0) {
    switch($op){
        case 'list':
            $blocks[0]['info']=t('Latest Chemical answers');
            $blocks[0]['cache']=BLOCK_NO_CACHE;
            $blocks[1]['info']=t('Post Chemical answers');
            $blocks[1]['cache']=BLOCK_NO_CACHE;
            return $blocks;
        case 'view':
            switch($delta){
                case 0:
                    $block['subject'] = t('Chemical answers');
                    $block['content'] = _question_answer_latest_questions();
                    break;
                case 1:
                    $block['subject'] = t('Post Chemical answers');
                    $block['content'] = _question_answer_latest_questions();
                    break;
            }
    }
    return $block;
}
```

后台去管理，操作这个block时候显示的标题

这个区块被放置到不同的区域
显示在网站时候的主题

要真正显示的内容

Hook_user – 处理与用户相关的操作

```
hook_user($op, &$edit, &$account, $category = NULL) {  
    switch ($op) {  
        case 'register': // 用户正在注册， 正在显示注册表单,我们增加一个表单元素  
            $fields[['age']] = array(  
                '#type' => 'textfield',  
                '#title' => t(你几岁了？'),  
            );  
            return $fields;  
        Break;  
        case 'login': // 用户正在登录,  
            drupal_set_message('Welcome to my website');  
        break;  
        case 'load': // 使用user_load的时候， 这里被调用到， 可以加载自己的数据，  
            $account->age = '26';  
        break;  
        case 'insert': // 用户刚刚注册成功， 用户数据被加到了数据库里  
            db_query("INSERT INTO {payuser} (uid, age .....  
        break;
```

Hook_user (cont.)

```
case 'logout': // 用户刚刚退出
    drupal_set_message('Welcome to back again;');
    break;
case 'view': // 用户帐号信息正在被查看
    $account->content='任意信息';
    break;
case 'category': // 用户分类信息的某个集合正在被请求
    $output_user[] = array(
        'name' => 'sinotech_user',
        //name就是指的分类的名字，，即$category的值
        'title' => t('sinotech Lists'),
        //而title就是分类的标题，也就是图1中显示的“sinotech Lists”。
        'weight' => 9);
    return $output_user;
    break;
```



注意：有的时候，我们需要获取当前用户的信息，往往使用的是
global \$user，
而hook_user里，使用的\$account,是被操作的账户用户。

Hook_nodeapi — 处理其他模块定义的node

这个钩子提供了一个可能，让你的模块去处理其他模块定义的node的一些操作，比如我们的例子里，模块定义了question 节点类型，但是我们想去操作“story”的生成，删除，就可以使用这个hook. [查看详细](#)

- hook_nodeapi(&\$node, \$op, \$a3 = NULL, \$a4 = NULL) {
switch (\$op) {
 case 'delete': // 当任意节点被删除的时候，这里的代码被执行
 if(\$node->type == "story") {
 }
 break;
 case 'insert': //当任意节点被插入到数据库时候，这里的代码也被执行。
 break;
 case 'load': //当 从数据库里读取数据的时候，这里的代码也被执行，我们可以加载额外的数据。
 break;
 case 'update': //当任意节点被修改的时候，数据里进行了update操作，这里的代码被执行
 .
 break;
 case 'view': //当任意节点被显示的时候，这里被执行，我们可以加载更多显示内容。
 break;

问题与解答



请把问题或者 [意见发送到 qq1392137041@gmail.com](mailto:qq1392137041@gmail.com), 或者论坛 www.chinaldap.com
或者直接了当加我qq: 1392137041, 或者直接进入 qqtalk房间号码: 6289868