Create Joomla! extensions - Manage the Back End- Part 2

Creating extensions for Joomla! 1.5 - Managing the Back End - Part 2 edit records

Writing extensions with **Marco's Component Maker for Joomla!** is quite easy since it is the program that takes care to write the skeleton for models, views and templates for editing. You have only to define the business logic for data managing, *but this is your own task.*

You need to install the example component created in previous article.

Back - End - Part 2 edit records

Versions 1.5+ generate full functional edit, but, of course, this is only a basic editing. You have to modify code for:

- add javaScript validation
- add JTable validation php code
- check data filtering validation in model
- manage 'Select List' values
- manage related table fields (foreign keys)

We will modify only the 'content' section of our test component, we don't care about the other (sections and categories) because modification are quite identical.

Add javaScript validation

The file generated by **Marco's Component Maker for Joomla!** includes java script skeleton for input validation :

see: */administrator/components/com_mcm/views/content/tpl/default.php*

```
<script type="text/javascript">
 function submitbutton(pressbutton) {
 var form = document.adminForm;
 if (pressbutton == 'cancel') {
 submitform( pressbutton );
 return;
 }
 // remove this code
 alert ('<?php echo 'Remember to add js check in ' . __FILE__ . ' after line n. '
. __LINE__; ?>');
 submitform( pressbutton );
 return;
 // end remove this code
 // do field validation
 if (form.My_Field_Name.value == "") {
 alert( "<?php echo JText::_( 'Field must have a name', true ); ?>" );
 } else if (form.My_Field_Name.value.match(/[a-zA-Z0-9]*/) !=
form.My_Field_Name.value) {
 alert( "<?php echo JText::_( 'Field name contains bad caracters', true ); ?>" );
 } else if
(form.My_Field_Name_typefield.options[form.My_Field_Name_typefield.selectedIndex].
value == "0") {
 alert( "<?php echo JText::_( 'You must select a field type', true ); ?>" );
 } else {
 submitform( pressbutton );
 }
 }
</script>
```

You have to complete it with the needed validation code. Remember to remove the alert between *//remove this code* markers.

Add JTable validation php code

You can add server side data consistency test modifying the php code generate for JTable derived class:

see: */administrator/components/com_mcm/tables/content.php*

Add the validation code in the *TableContent->check()*; return false on fault,

```
function check(){
 // write here data validation code
 if(!$this->fulltext){
 $this->setError( 'Articles must have text!' );
 return false;
 }

 return parent::check();
 }
```

The check() method is called by the store() method in model, invoked by the save() method in controller.

Check data filtering in model

Data submitted from editing form are collected and filtered in McmModelContent->store() method in file:
/administrator/components/com_mcm/models/content.php

```
// mcm code
 $data['id'] = JRequest::getVar('id', '', 'post', 'int');
 $data['introtext'] = JRequest::getVar('introtext', '', 'post', 'string',
JREQUEST_ALLOWRAW);
 $data['fulltext'] = JRequest::getVar('fulltext', '', 'post', 'string',
JREQUEST_ALLOWRAW);
 // ... omissis ...
// mcm code
```

Usually you do not have to change this code, but remember to update it if you add input fields to the form. Please note also the needed validation to accept HTML code.

Manage 'Select List' values

When you set the 'Render As' field configuration to 'Select List' additional code is written into model and view entry point. This is useful for 'set'and 'enum' fields or when list values is not store in a DBase table.

Model File

Methods are written for every selected fields with Joomla! naming convention as *getFiedname()*.

see: */administrator/components/com_mcm/models/content.php*

We will modify only the getAccess() method, because methods to get category and section (getSectionid(), getCatid()) will not be used: are related tables, not fixed list values.

```
public function &getAccess(){
 $options = array(
 JHTML::_('select.option', '0', 'Free for All' ),
 JHTML::_('select.option', '1', 'Registered users' ),
 JHTML::_('select.option', '2', 'Administrative users' )
 );
 return $options;
}
```

View entry point

Method declared in model are invoked in view and data are stored in the *$dataOptions* associative array.
see: */administrator/components/com_mcm/views/content/view.html.php*

```
// create options for 'select' used in template

$dataOptions = array();
foreach(explode(',', 'sectionid,catid,access') as $field){
 if (!$field) continue;
 //options array are generated in the model...
 $dataOptions[$field] =& $this->get( ucfirst($field) );
}
```

You can remove *sectionid* and *catid* from filed list or rewrite the code as

```
 $dataOptions['access'] =& $this->getAccess;
```

$dataOptions is stored as reference and values are retrieved in the template to populate select.

Manage related table fields (foreign keys)

To manage foreign keys you have to:

- check the option 'Get Field Data' in the table from which you want to get the data
- select 'Select List' in the 'Render As' for the foreign key

We selected the 'Get Field Data' for the 'title' field in jos_sections and jos_categories then 'Select List' for 'sectionid' and 'catid' in jos_content table.

'Get Field Data' generates method for obtaining the key-value pair (primary key - field) from the table. Methods are written in models for recordset listing:

Please pay attention!

'Get Field Data' doesn't check if method name is valid. so if you select the field 'name' will be generated the getName() method overriding the JModel->getName(). In doubt add a suffix to auto-generated method!

Getting data from models

see: */administrator/components/com_mcm/models/sectionslist.php*
see: */administrator/components/com_mcm/models/ `categorieslist.php*

Methods are identical, because field names are identical too.

```
public function &getTitle(){
 $db =& JFactory::getDBO();
 $db->setQuery( 'SELECT `id` AS value, `title` AS text FROM `#__categories` ORDER
BY name');

 $options = array();
 foreach( $db->loadObjectList() as $r){
 $options[] = JHTML::_('select.option', $r->value, $r->text );
 }
 return $options;
}
```

Method returns an array of options for populating select box.

Add models to controller

Now we have to include the two models into controller before to call methods.
Please pay attention tho modify the right controller: is the controller for managing the recordset list NON the single record.

see: */administrator/components/com_mcm/controllers/contentlist.php*

modify the edit() method adding before *parent::display();*

```
// Related table model include [NB: include recordset list model]
$altModel1 =& $this->getModel('sectionslist');
$altModel2 =& $this->getModel('categorieslist');

$view =& $this->getView('content', 'html');
$view->setModel($altModel1);
$view->setModel($altModel2);
```

Example code is present (commented).

Show related data in view

Last step: invoking the methods from view:

see: */administrator/components/com_mcm/views/content/view.html.php*

modify the example code:

```
$rmodel =& $this->getModel('sectionslist');
$dataOptions['sectionid'] =& $rmodel->getTitle();
$rmodel =& $this->getModel('categorieslist');
$dataOptions['catid'] =& $rmodel->getTitle();
```

Now when you go in editing mode you will see sections and categories names instead of their id

(foreign key), but on submit the id will be sent.

Manage related table fields (the wrong way)

I'm a old C developer and I know there is a 'long & good' way and a 'quick & dirty' one to do a work.

Alternative step: do it dirty:

see: */administrator/components/com_mcm/views/content/view.html.php*

modify the example code:

```
$dataOptions['sectionid'] =& JHTML::_('list.genericordering', 'SELECT id AS
value, title AS text FROM #__sections ORDER BY title');
$dataOptions['catid'] =& JHTML::_('list.genericordering', 'SELECT id AS value,
title AS text FROM #__categories ORDER BY title');
```

Quick but wrong... It's not a *best practices* to:

- put SQL code in the view
- do not get data from the right model/table object (we are in content, not in categories/sections).
- and so on...

The end

Ok, now I think you are able to write your own more useful components.

bye,
marco

Commenti

#3 **RODOLFO SEALES** 2011-12-12 03:21 **0**

Mi congratulo in anticipo lo sviluppo di questo meraviglioso strumento per Joomla e vorrei sapere se è possibile inserire sul tuo sito un esempio completo ben spiegato, passo dopo passo, come si sviluppa un'applicazione con la MCM, a partire fine, spiegando le istruzioni e il loro effetto, come potrebbe essere esteso ad altre funzioni, ecc,,, si spera compreso l'accesso al database, uno stile di guida, potrebbe essere per un video o di forma scritta. Un buon algoritmo potrebbe essere una estensione per l'uso in classe A note scuola per soggetto, ecc ... ecc, che accede a questo database, scrivendo al database, i dati di cancellare, modificarlo, ecc ... Apprezzo la vostra cortese attenzione a questo.

Citazione

#2 **RODOLFO SEALES** 2011-12-12 03:19 **0**

De antemano quiero felicitarlo por el desarrollo de esta maravillosa herramienta para JOOMLA y me gustaria saber si para usted es posible colocar en su sitio web un ejemplo completo muy bien explicado, paso a paso, de como se desarrolla una aplicacion con el MCM, de comienzo a final, explicando las instrucciones y su efecto, como podria ampliarse para otras funciones, etc,,,ojala incluyendo el acceso a la base de datos, a estilo de tutorial,

podria ser por un video o de forma escrita. Un buen algoritmo podria ser una extension para utilizarla en la calificacion de notas de un colegio por asignaturas, etc... etc, que con esto se accede a bases de datos, escritura a la base, borrar datos, modicificarlo,,etc...

Agradezco su amable atencion a la presente.

Citazione

#1 **RODOLFO SEALES** 2011-12-12 03:19 **0**

Hello Marco, how are you?
I congratulate in advance the development of this wonderful tool for Joomla and I would like to know if you can place on your website a complete example well explained, step by step, how you develop an application with the MCM, starting end, explaining the instructions and their effect, as might be extended to other functions, etc,,, hopefully including access to the database, a tutorial style, could be for a video or written form. A good algorithm could be an extension for use in rating a school notes by subject, etc ... etc., that accesses this database, writing to the database, delete data, modicificarlo, etc ... I appreciate your kind attention to this.

Citazione

Aggiorna elenco commenti

RSS feed dei commenti di questo post.

Aggiungi commento

E-Mail

Sito web

1000 caratteri rimasti

Aggiorna

Invia

JComments