11
Panels and Widgets – Part Two

In the previous chapter, we built the basics of the panels and widgets system. You can now create panels, can drag widgets into them, and can see the results on the front-end of the site.

In this chapter, we will enhance the system letting you customize the widgets and letting you choose what pages the panels and widgets are visible on.

Widget forms

In the last chapter, we had widgets showing on the front-end, but with a default **This Content Snippet is not yet defined** message.

In order to change the message to something more useful, we will need to do more work in the admin area.

Firstly, when you reload the Panels area of the admin, you'll see that our right panel widget appears to have vanished. We've recorded it in the database, but have not set the page to show it on loading.

Edit `/ww.plugins/panels/admin/js.js` and where the `.panel-opener` is added to the `h4` of `.panel-wrapper`, add these highlighted lines:

```
widgets_container.appendTo(panel);
var widgets=panel.data('widgets');
for(var i=0;i<widgets.length;++i){
  var p=widgets[i];
  var w=buildRightWidget(p);
  w.appendTo(widgets_container);
  if(p.header_visibility)
```

```
            $('input.widget_header_visibility',w)[0]
              .checked=true;
        }
        $('<br style="clear:both" />').appendTo(panel);
```

When we created the panels in the JavaScript, we set the wrapper's `data('widgets')` with data from the database. This code takes that data and adds the widgets on-the-fly, when the panel is opened.

Because each widget is different, we need to provide a way to load up external configuration forms.

To do this, replace the `showWidgetForm()` stub function with this:

```
function showWidgetForm(w){
  if(!w.length)w=$(this).closest('.widget-wrapper');
  var f=$('form',w);
  if(f.length){
    f.remove();
    return;
  }
  var form=$('<form></form>').appendTo(w);
  var p=w.data('widget');
  if(ww_widgetForms[p.type]){
    $('<button style="float:right">Save</button>')
      .click(function(){
        w.find('input,select').each(function(i,el){
          p[el.name]=$(el).val();
        });
        w.data('widget',p);
        updateWidgets(form.closest('.panel-wrapper'));
        return false;
      })
      .appendTo(form);
    var fholder=$('<div style="clear:both;border-bottom:1px solid
#416BA7">loading...</div>').prependTo(form);
    p.panel=$('h4>span.name',form.closest('.panel-wrapper')).eq(0).
text();
    fholder.load(ww_widgetForms[p.type],p);
  }
  else $('<p>automatically configured</p>').appendTo(form);
  $('<a href="javascript:;" title="remove widget">remove</a>')
    .click(function(){
      if(!confirm('Are you sure you want to remove this widget from
this panel?'))return;
```

```
        var panel=w.closest('.panel-wrapper');
        w.remove();
        updateWidgets(panel);
      })
      .appendTo(form);
    $('<span>, </span>').appendTo(form);
    $('<a href="javascript:;">visibility</a>')
      .click(widget_visibility)
      .appendTo(form);
    $('<span>, </span>').appendTo(form);
    $('<a class="disabled" href="javascript:;">'+(p.disabled?'disabled':
  'enabled')+'</a>')
      .click(widget_toggle_disabled)
      .appendTo(form);
  }
  function widget_toggle_disabled(){}
  function widget_visibility(){}
```

The first few lines toggle the form closed, if it's already closed.

Then, if a form is provided (we'll get to this in a moment), it is added to a form element, along with a **Save** button. The external form is embedded in the element using jQuery's `.load()` function, which also runs any scripts in the external form.

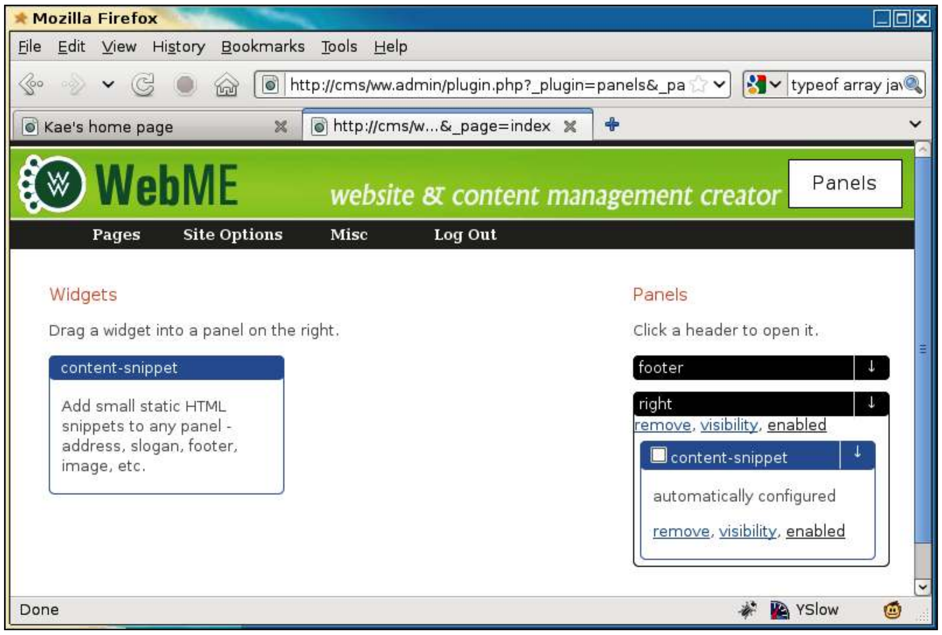
If no form is provided, a message is shown saying that the widget is automatically configured.

Then, a number of links are added to let you remove, disable, or set the pages that the widget is active on.

The only link that works at the moment is the remove link, which simply deletes the widget element, and then updates the panel.

The other links are served by stub functions. We will finish them all before the end of the chapter!

Here's a screenshot showing the widget as it appears now:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_pa
typeof array ja
Kae's home page
http://cms/w...&_page=index
WebME
website & content management creator
Panels
Pages Site Options Misc Log Out
Done
YSlow

Widgets

Drag a widget into a panel on the right.

content-snippet

Add small static HTML snippets to any panel - address, slogan, footer, image, etc.

Panels

Click a header to open it.

footer

right

remove, visibility, enabled

content-snippet

automatically configured

remove, visibility, enabled

When we were initially creating `/ww.plugins/panels/admin/index.php`, we added a simple "widget forms" comment block. Now, replace that with this:

```
// { widget forms
echo 'ww.widgetForms={';
$ws=array();
foreach($PLUGINS as $n=>$p){
  if(isset($p['admin']['widget'])
      && isset($p['admin']['widget']['form_url']))
    $ws[]='"'.$n.'":"'
      .addslashes($p['admin']['widget']['form_url']).'"';
}
echo join(',',$ws);
echo '};';
// }
```

This builds up a list of panel forms and echoes them to the browser's HTML:

```
ww_widgets=[{type:"content-snippet",description:"Add small static HTML
snippets to any panel - address, slogan, footer, image, etc."}];
ww_widgetForms={"content-snippet":"/ww.plugins/content-snippet/admin/
widget-form.php"};
</script>
```

So let's create the form. Make the `/ww.plugins/content-snippet/admin` directory and add a `widget-form.php` file to it:

```
<?php
require $_SERVER['DOCUMENT_ROOT'].'/ww.admin/admin_libs.php';
// { return content from table if requested
if(isset($_REQUEST['get_content_snippet'])){
  require '../frontend/index.php';
  $o=new stdClass();
  $o->id=(int)$_REQUEST['get_content_snippet'];
  $ret=array('content'=>content_snippet_show($o));
  echo json_encode($ret);
  exit;
}
// }
// { set ID and show link in admin area
if(isset($_REQUEST['id']))$id=(int)$_REQUEST['id'];
else $id=0;
echo '<a href="javascript:content_snippet_edit('.$id.');"
  id="content_snippet_editlink_'.$id.'"
  class="content_snippet_editlink">view or edit snippet</a>';
// }
?>
```

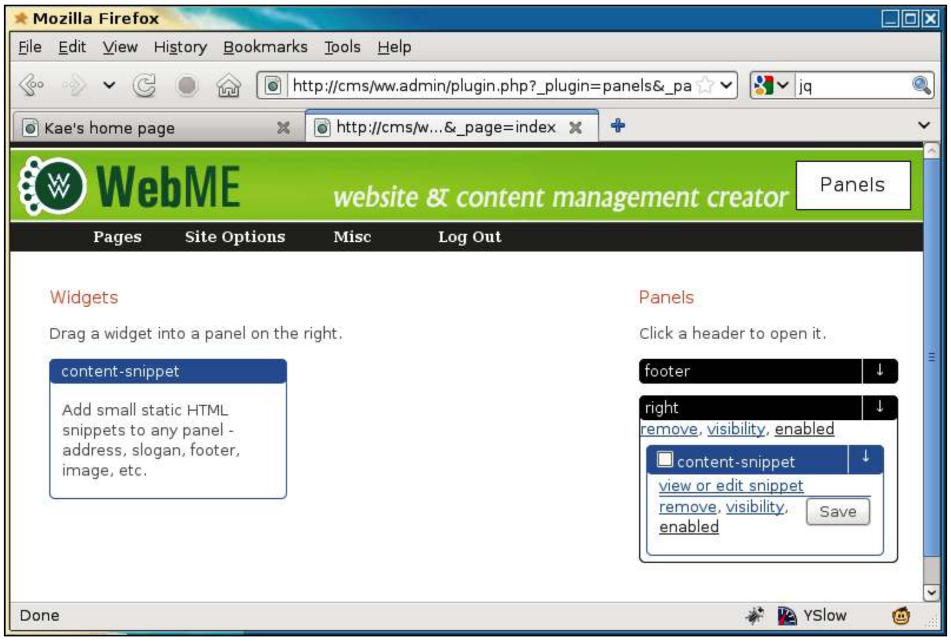
This is not the end of the file—we've only just started. It's a large snippet, so I'll explain just this bit first.

First, we check that the user is an admin and load up the admin libraries, in case they're needed.

Then, we check to see if the browser has asked for the content of the snippet. There'll be more on that in a bit when we talk about the JavaScript part of the file.

Next, we make sure that an ID has been provided or else set it to 0.

Finally, we output a link for the browser to show to the user.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_pa
jq
Kae's home page
http://cms/w...&_page=index
WebME
website & content management creator
Panels
Pages
Site Options
Misc
Log Out
Done
YSlow

Widgets

Drag a widget into a panel on the right.

content-snippet

Add small static HTML snippets to any panel - address, slogan, footer, image, etc.

Panels

Click a header to open it.

footer

right

remove, visibility, enabled

content-snippet

view or edit snippet

remove, visibility, enabled

Save

So next, we need to define what happens when the **view or edit snippet** link is clicked.

Add the following code to the same file:

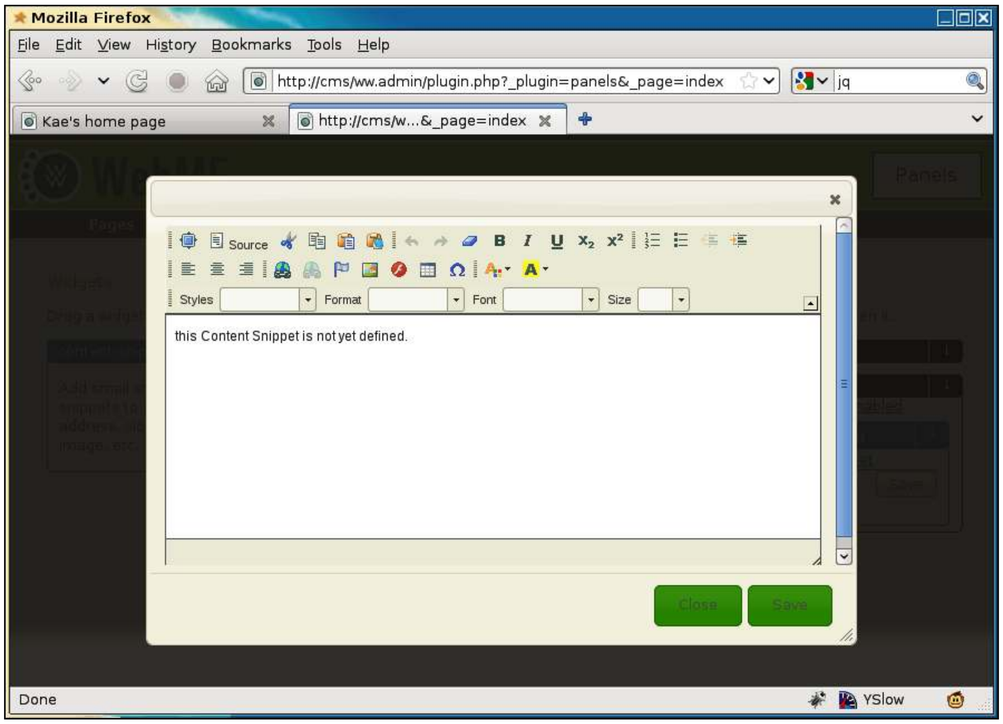
```
<script>
if(!window.ww_content_snippet)window.ww_content_snippet={
  editor_instances:0
};
function content_snippet_edit(id){
  var el=document.getElementById('content_snippet_editlink_'
    +id);
  ww_content_snippet.editor_instances++;
  var rtenum=ww_content_snippet.editor_instances;
  var d=$('<div><textarea style="width:600px;height:300px;" '
    +'id="content_snippet_html'+rtenum+'" '
    +'name="content_snippet_html'+rtenum+'"></textarea>'
    +'</div>');
  $.getJSON(
    '/ww.plugins/content-snippet/admin/widget-form.php',
    {'get_content_snippet':id},
    function(res){
      d.dialog({
          minWidth:630,
          minHeight:400,
          height:400,
          width:630,
          modal:true,
          beforeclose:function(){
            if(!ww_content_snippet.rte)return;
            ww_content_snippet.rte.destroy();
            ww_content_snippet.rte=null;
          },
          buttons:{
            'Save':function(){
              // leave empty for now
            },
            'Close':function(){
              d.remove();
            }
          }
        });
      ww_content_snippet.rte=CKEDITOR.replace(
        'content_snippet_html'+rtenum,
        {filebrowserBrowseUrl:"/j/kfm/",menu:"WebME"}
      );
      ww_content_snippet.rte.setData(res.content);
    });
  }
  </script>
```

What this does is create a dialog, add an instance of the CKeditor rich text editor to it, and then request the snippet content from the server (see the previous PHP section for that).

Note the steps we've taken with CKeditor. At the time of writing, CKeditor 3 is still not complete—the documentation on the main website, for example, has hardly anything in the JavaScript section.

Destroying an instance of CKeditor dynamically is still not absolutely safe, so what we do is that when a dialog is closed, we do what we can using the `.destroy()` method provided by CKeditor. To be extra sure, we don't reuse a destroyed instance, but we always initiate a new one (see the use of `editor_instances` in the code).

The previous block of code will render this to the screen when the **view or edit snippet** link is clicked:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_page=index
Kae's home page
http://cms/w...&_page=index
Source
Styles Format Font Size
Close Save
Done
YSlow

this Content Snippet is not yet defined.

You can now insert any HTML you want into that.

Saving the snippet content

The code I've shown doesn't include the **Save** function. Let's add that now.

Edit the JavaScript part of the file, and replace the line `// leave empty for now` with this:

```
var html=ww_content_snippet.rte.getData();
$.post('/ww.plugins/content-snippet/admin/'
    +'widget-form.php',
  {'id':id,'action':'save','html':html},
  function(ret){
    if(ret.id!=ret.was_id){
      el.id='content_snippet_editlink_'+ret.id;
      el.href='javascript:content_snippet_edit('
        +ret.id+')';
```

```
                }
                id=ret.id;
                var w=$(el).closest('.widget-wrapper');
                var wd=w.data('widget');
                wd.id=id;
                w.data('widget',wd);
                updateWidgets(w.closest('.panel-wrapper'));
                d.remove();
              },
              'json'
            );
```

The **Save** functionality sends the RTE contents to the server.

On the server, we will save it to the database. If an ID was provided, it will be an update; otherwise, it will be an insert.

In either case, data including the ID is then returned to the client. If the database entry was an insert, the widget is updated with the new ID.

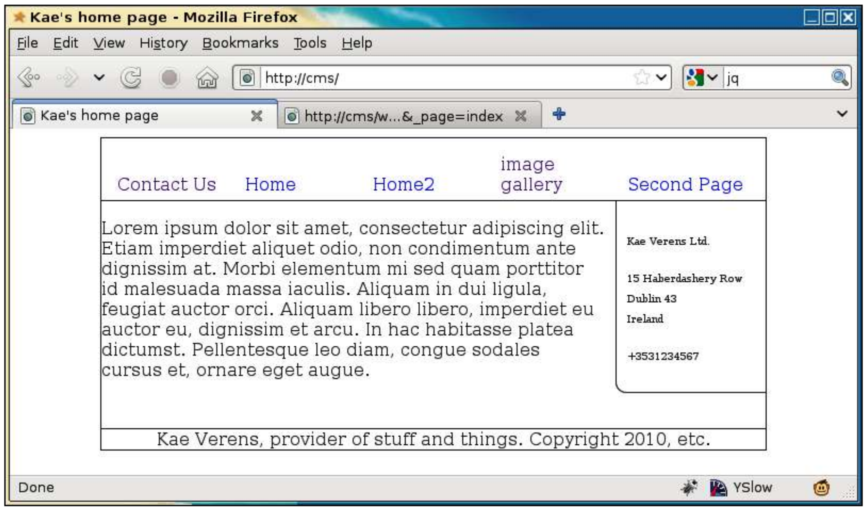
The panel is then saved to the server in case any of its data (for instance, the widget data in the case where an ID was created) was changed.

In the PHP section, add this to the top of the file, below the require line:

```
// { save data to the database if requested
if(isset($_REQUEST['action']) && $_REQUEST['action']=='save'){
  $id=(int)$_REQUEST['id'];
  $id_was=$id;
  $html=addslashes($_REQUEST['html']);
  $sql="content_snippets set html='$html'";
  if($id){
    $sql="update $sql where id=$id";
    dbQuery($sql);
  }
  else{
    $sql="insert into $sql";
    dbQuery($sql);
    $id=dbOne('select last_insert_id() as id','id');
  }
  $ret=array('id'=>$id,'id_was'=>$id_was);
  echo json_encode($ret);
  exit;
}
// }
```

This simply saves the offered data into the database and returns the original ID as submitted, along with the new one, in case an insert was made and the client-side code needs to update itself.

With this code in place, you can now edit your content snippets, as can be seen in the following screenshot:


Kae's home page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/
jq
Kae's home page
http://cms/w...&_page=index
Contact Us
Home
Home2
image gallery
Second Page
Done
YSlow

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam imperdiet aliquet odio, non condimentum ante dignissim at. Morbi elementum mi sed quam porttitor id malesuada massa iaculis. Aliquam in dui ligula, feugiat auctor orci. Aliquam libero libero, imperdiet eu auctor eu, dignissim et arcu. In hac habitasse platea dictumst. Pellentesque leo diam, congue sodales cursus et, ornare eget augue.

Kae Verens Ltd.

15 Haberdashery Row
Dublin 43
Ireland

+3531234567

Kae Verens, provider of stuff and things. Copyright 2010, etc.

You can see that I've used the panel on the right-hand side to add an address and the bottom panel to add some standard footer-type stuff.

You can add as many widgets to a panel as you want.

We are now finished with the Content Snippet plugin.

Renaming widgets

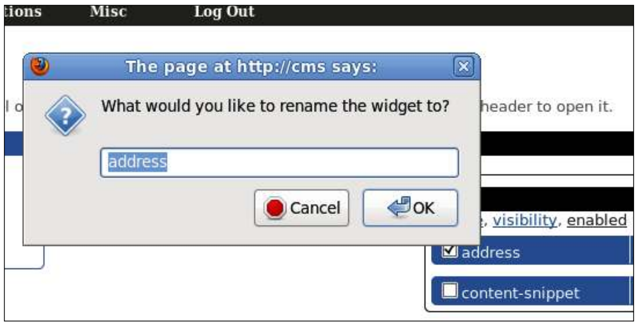
In a busy website, it is a nuisance if you have a load of widgets in panels and are not sure which one you're looking for. For example, if one of the Content Snippet widgets is an address, it is better that it says **address** in the panel, than **content snippet**.

We've already added the code that calls the function `widget_rename()`, when the widget header is clicked.

Replace the stub `widget_rename()` function in `/ww.plugins/panels/admin/js.js` with this:

```
function widget_rename(ev){
  var h4=$(ev.target);
  var p=h4.closest('.widget-wrapper').data('widget');
  var newName=prompt('What would you like to rename the '
    +'widget to?',p.name||p.type);
  if(!newName)return;
  p.name=newName;
  h4.closest('.widget-wrapper').data('widget',p);
  updateWidgets($(h4).closest('.panel-wrapper'));
  h4.text(newName);
}
```

Very simply put, this sets `p` to the current widget data, asks for a new name for the widget, sets `p.name` equal to that new name, and then saves the widget data again.



What would you like to rename the widget to?

address

In small sites where there're only a few widgets in use, this may seem like overkill, but in more complex sites, this is necessary.

Widget header visibility

On the front-end, you might want to have **Address** written above the address section of the panel on the right.

For Content Snippets, this is simple, as you just need to add it to the HTML that you're already building.

But, if you're using a different widget, then there may not be an editable section of HTML. For example, in an RSS reader widget, which simply displays a list of items from an RSS stream, it would be overkill to add an editor for user-controlled HTML.

So, for these cases, we provide a check-box in the widget's header which lets you tell the server to add a `<h4>` element before the widget is rendered.

Add this code to the `js.js` file, replacing the `widget_header_visibility()` stub function:

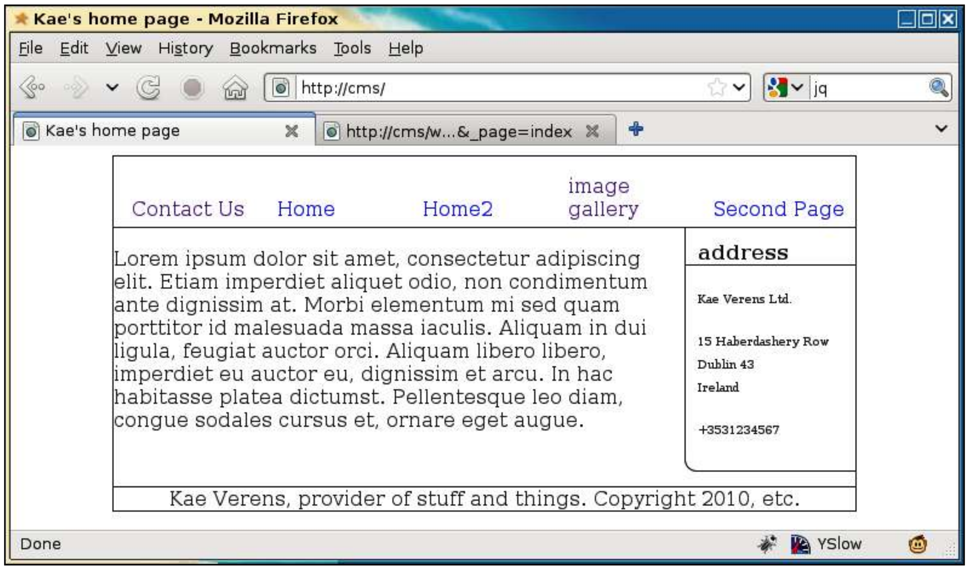
```
function widget_header_visibility(ev){
  var el=ev.target,vis=[];
  var w=$(el).closest('.widget-wrapper');
  var p=w.data('widget');
  p.header_visibility=el.checked;
  w.data('widget',p);
  updateWidgets(w.closest('.panel-wrapper'));
}
```

Similar to the widget renaming section, this simply edits the widget data, saves it, and updates the visuals.

For the front-end, edit `/ww.plugins/panels/plugin.php` and add the following highlighted line:

```
  foreach($widgets->widgets as $widget){
    if(isset($widget->header_visibility)
        && $widget->header_visibility)
      $h.='<h4 class="panel-widget-header '
        .preg_replace('/[^a-z0-9A-Z\-]/','',$widget->name)
        .'">'.htmlspecialchars($widget->name).'</h4>';
    if(isset($PLUGINS[$widget->type])){
      if(isset($PLUGINS[$widget->type]['frontend']['widget'])){
```

This will add a `<h4>` before any widgets you select:


Kae's home page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://cms/

Kae's home page | http://cms/w...&_page=index

Contact Us Home Home2 image gallery Second Page

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam imperdiet aliquet odio, non condimentum ante dignissim at. Morbi elementum mi sed quam porttitor id malesuada massa iaculis. Aliquam in dui ligula, feugiat auctor orci. Aliquam libero libero, imperdiet eu auctor eu, dignissim et arcu. In hac habitasse platea dictumst. Pellentesque leo diam, congue sodales cursus et, ornare eget augue.

address

Kae Verens Ltd.

15 Haberdashery Row
Dublin 43
Ireland

+3531234567

Kae Verens, provider of stuff and things. Copyright 2010, etc.

Done YSlow

Notice the **address** header. Also, notice that I've not added a header to the footer panel.

Disabling widgets

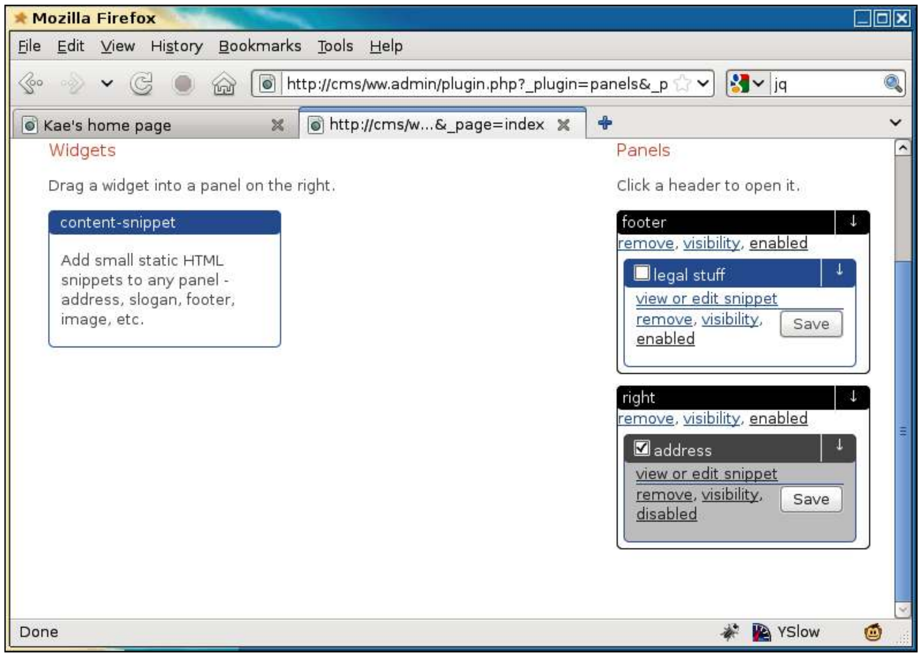
If you have a number of different widgets that you use for different occasions, it is useful to disable those that you are not currently using, so they don't appear in the front-end and yet are available to re-enable at any point.

An example of this might be a menu for a restaurant, where the "soup of the day" revolves based on the day. If there are seven different soups, it's a simple matter to disable the six that you are not displaying, and each day, enable the next and disable the previous.

As you can imagine, the admin side code of this is easy, based on the most recent examples. Edit the `admin/js.js` file and replace the `widget_toggle_disabled()` stub function with this:

```
function widget_toggle_disabled(ev){
  var el=ev.target,vis=[];
  var w=$(el).closest('.widget-wrapper');
  var p=w.data('widget');
  p.disabled=p.disabled?0:1;
  w.removeClass().addClass('widget-wrapper '
    +(p.disabled?'disabled':'enabled'));
  $('.disabled',w).text(p.disabled?'disabled':'enabled');
  w.data('widget',p);
  updateWidgets(w.closest('.panel-wrapper'));
}
```

In the admin area, after disabling a panel, here's how it looks:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_p
jq
Kae's home page
http://cms/w...&_page=index
Widgets
Drag a widget into a panel on the right.
content-snippet
Add small static HTML snippets to any panel - address, slogan, footer, image, etc.
Panels
Click a header to open it.
footer
remove, visibility, enabled
legal stuff
view or edit snippet
remove, visibility, enabled
Save
right
remove, visibility, enabled
address
view or edit snippet
remove, visibility, disabled
Save
Done
YSlow

And on the front-end, we simply return a blank string if the panel is disabled.

To do this, edit the `plugin.php` and add the highlighted line:

```
foreach($widgets->widgets as $widget){
  if(isset($widget->disabled) && $widget->disabled)continue;
  if(isset($widget->header_visibility)
      && $widget->header_visibility)
    $h.='<h4 class="panel-widget-header '
      .preg_replace('/[^a-z0-9A-Z\-]/','',$widget->name)
      .'">'.htmlspecialchars($widget->name).'</h4>';
```

If the widget is disabled, you simply ignore that widget and carry onto the next iteration of the loop.

Disabling a panel

Panels are not recorded the same way as widgets, so there's slightly more work needed.

The JavaScript is basically the same. We already have the links for **remove**, **visibility**, and **enable/disable** in there, so it's just a matter of adding the functions they call.

Add this function to `/ww.plugins/panels/admin/js.js`:

```
function panel_toggle_disabled(i){
  var p=ww_panels[i];
  p.disabled=p.disabled?0:1;
  var panel=$('#panel'+p.id);
  panel
    .removeClass()
    .addClass('panel-wrapper '
      +(p.disabled?'disabled':'enabled'));
  $('.controls .disabled',panel)
    .text(p.disabled?'disabled':'enabled');
  ww_panels[i]=p;
  $.get('/ww.plugins/panels/admin/save-disabled.php?id='
    +p.id+'&disabled='+p.disabled);
}
```

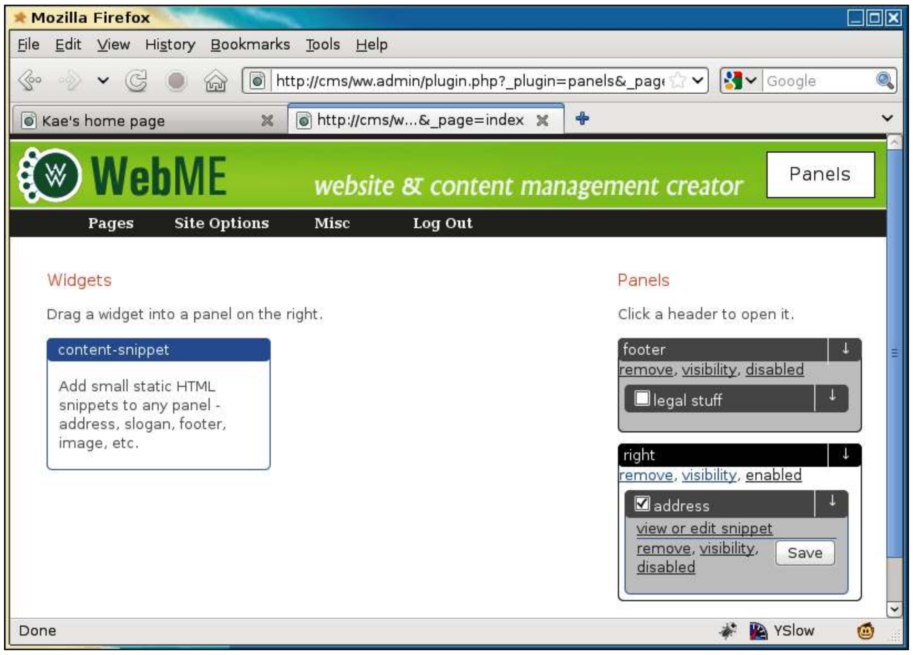
This function switches classes and the visible text in the panel to toggle its mode between enabled and disabled, then calls a server-side file to save the state.

Create the file `/ww.plugins/panels/admin/save-disabled.php` to handle the saving:

```
<?php
require $_SERVER['DOCUMENT_ROOT'].'/ww.admin/admin_libs.php';
if(isset($_REQUEST['id']) && isset($_REQUEST['disabled'])){
  $id=(int)$_REQUEST['id'];
  $disabled=(int)$_REQUEST['disabled'];
  dbQuery("update panels set disabled='$disabled' where id=$id");
}
echo 'done';
```

After the **enabled** link is clicked (below the panel name), the entire panel is then disabled and is not visible from the front-end.

In the admin area, this is indicated by graying out the entire panel:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_page
Google
Kae's home page
http://cms/w...&_page=index
WebME
website & content management creator
Panels
Pages Site Options Misc Log Out
Widgets
Drag a widget into a panel on the right.
content-snippet
Add small static HTML snippets to any panel - address, slogan, footer, image, etc.
Panels
Click a header to open it.
footer
remove, visibility, disabled
legal stuff
right
remove, visibility, enabled
address
view or edit snippet
remove, visibility, disabled
Save
Done
YSlow

Because the widget statuses are not changed, re-enabling the panel by clicking on **disabled** will bring the panel back to the exact status it had before. For example., if half the widgets contained in it were disabled, then the exact same widgets are disabled and the rest are enabled.

Deleting a panel

Panels are tied in with templates. The panels that appear in the admin area depend on what is visible in the front-end.

Let's say that you were using one theme which had a footer panel and a right panel. And then, you switch to a new theme which has a header, left panel, and footer.

Loading a page in the front-end will register the new header and left panels, but will not remove the obsolete right panel.

Panels don't automatically delete when you switch themes. Because we have not tied a Smarty parser into the panels system, the CMS does not automatically know if a panel is no longer needed because it is not in the new skin.

Add the following function to `admin/js.js`:

```
function panel_remove(i){
  var p=ww_panels[i];
  var id=p.id;
  if(!confirm('Deleting this panel '
    +'will remove the configurations of its contained '
    +'widgets. Are you /sure/ you want to remove this? Note '
    +'that your panel will be recreated (without its '
    +'widgets) if the site theme has it defined.'))return;
  $.get('/ww.plugins/panels/admin/remove-panel.php?id='+id,
    function(){
      $('#panel'+id).remove();
    });
}
```

In this function, we first get the panel ID.

Next, we verify that the admin means to delete the panel—once it is deleted, the widget data will also be deleted, so it's important that the admin realizes this.

Finally, we send to the server to do the deletion, and remove the panel element from the page.

Here is the server-side file, `/ww.plugins/panels/admin/remove-panel.php`:

```
<?php
require $_SERVER['DOCUMENT_ROOT'].'/ww.admin/admin_libs.php';
if(isset($_REQUEST['id'])){
  $id=(int)$_REQUEST['id'];
  dbQuery("delete from panels where id=$id");
}
echo 'ok';
```

Again, a very simple file.

We first check that the requester is an admin, then remove the table row that corresponds to the panel ID.

Panel page visibility—admin area code

Sometimes, you will want a panel to appear on only a few pages.

For example, you may have a vertical side panel that you want to appear on all pages, except for one or two pages which need a lot of space, so you want to hide the panel on those pages.

Instead of creating different templates for these pages, you could simply hide the panels.

To do this, we first need to select what pages the panel is visible on.

Add the following function to `admin/js.js`:

```
function panel_visibility(id){
  $.get('/ww.plugins/panels/admin/get-visibility.php',
      {'id':id},function(options){
    var d=$('<form><p>This panel will be visible in <select '
      +'name="panel_visibility_pages[]" multiple="multiple">'
      +options+'</select>. If you want it to be visible in '
      +'all pages, please choose <b>none</b> to indicate '
      +'that no filtering should take place.</p></form>');
    d.dialog({
      width:300,
      height:400,
      close:function(){
        $('#panel_visibility_pages').remove();
        d.remove();
      },
      buttons:{
        'Save':function(){
          var arr=[];
          $('input[name="panel_visibility_pages[]"]:checked')
            .each(function(){
              arr.push(this.value);
            });
          $.get('/ww.plugins/panels/admin/save-visibility'
            +'.php?id='+id+'&pages='+arr);
          d.dialog('close');
        },
        'Close':function(){
          d.dialog('close');
        }
      }
    });
  });
}
```

This function is quite large compared to the previous functions, but it is also more complex.

In this case, we first retrieve the list of pages already selected from the server and then show this to the admin.

The admin selects which pages the panel should be visible on. Click on **Save**.

This then gets the page IDs of the selected options and saves this to the server.

There are two server-side files to create. First, the file to retrieve the list of pages is `/ww.plugins/panels/admin/get-visibility.php`:

```
<?php
require $_SERVER['DOCUMENT_ROOT'].'/ww.admin/admin_libs.php';
function panel_selectkiddies($i=0,$n=1,$s=array(),$id=0,$prefix=''){
  $q=dbAll('select name,id from pages where parent="'.$i
    .'" and id!="'.$id.'" order by ord,name');
  if(count($q)<1)return;
  $html='';
  foreach($q as $r){
    if($r['id']!=''){
      $html.='<option value="'.$r['id'].'" title="'
        .htmlspecialchars($r['name']).'"';
      $html.=(in_array($r['id'],$s))
        ?' selected="selected">':'>';
      $name=strtolower(str_replace(' ','-',$r['name']));
      $html.= htmlspecialchars($prefix.$name).'</option>';
      $html.=panel_selectkiddies($r['id'],$n+1,$s,$id,
        $name.'/');
    }
  }
  return $html;
}
$s=array();
if(isset($_REQUEST['id'])){
  $id=(int)$_REQUEST['id'];
  $r=dbRow("select visibility from panels where id=$id");
  if(is_array($r) && count($r)){
    if($r['visibility'])$s=json_decode($r['visibility']);
  }
}
if(isset($_REQUEST['visibility']) && $_REQUEST['visibility']){
  $s=explode(',',$_REQUEST['visibility']);
}
echo panel_selectkiddies(0,1,$s,0);
```

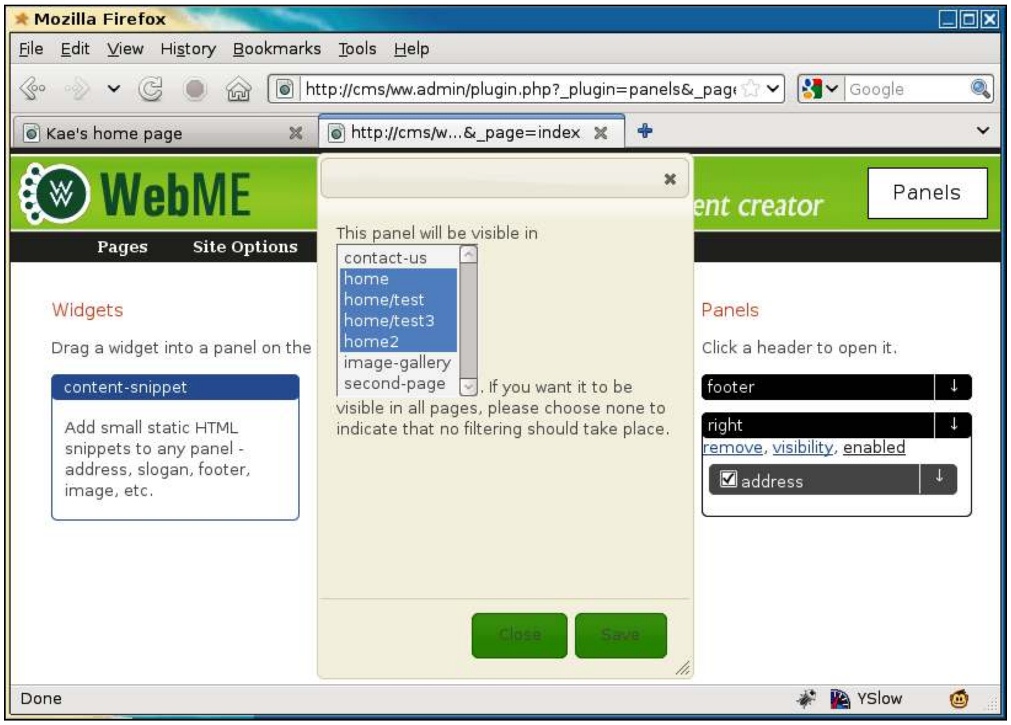
First, we make sure (as always) that the request came from an admin.

Next, we define the `panel_selectkiddies()` function that builds up an option list composed of pages in an hierarchical tree fashion.

Finally, we retrieve the list of pages that are already selected and display the options using that list, to mark some as selected.

> This could easily be added as a function of the core engine.
>
> The main reason it is not currently in the core engine is that we have one single use case for it and the core functions should really be functions that are used multiple times.
>
> If you find yourself rewriting functionality that exists in other plugins, then that functionality should be re-factored and added to the core.

When we click **visibility** under the panel, this is what we get:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_page
Google
Kae's home page
http://cms/w...&_page=index
WebME
ent creator
Panels
Pages
Site Options
Widgets
Drag a widget into a panel on the
content-snippet
Add small static HTML snippets to any panel - address, slogan, footer, image, etc.
Panels
Click a header to open it.
footer
right
remove, visibility, enabled
address
Close
Save
Done
YSlow

This panel will be visible in

contact-us
home
home/test
home/test3
home2
image-gallery
second-page

. If you want it to be visible in all pages, please choose none to indicate that no filtering should take place.

Not very user-friendly, is it?

We can improve this by using the jQuery inlinemultiselect plugin, available from here: `http://code.google.com/p/inlinemultiselect/`.

The inlinemultiselect plugin, by Peter Edwards, is an enhancement of some work I did a few years beforehand to make multi-select elements easier to use.

The version of the file that I'm using is a slight enhancement of the version on the Google repository. I've submitted my changes back and am waiting for the changes to be added to the repository.

In the meantime, you can get my version from Packt, by downloading the code for this chapter.

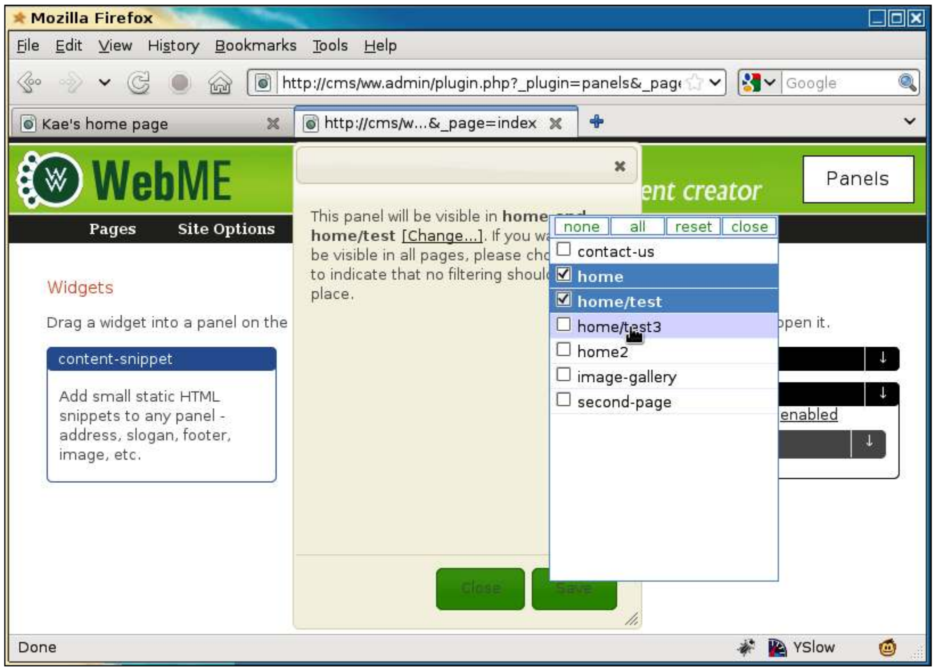
I place the file as `/ww.plugin/panels/admin/jquery.inlinemultiselect.js` and then edit the file `/ww.plugin/panels/admin/index.php` to link it in (highlighted lines):

```
?>
</script>
<script src="/ww.plugins/panels/admin/js.js"></script>
<script src="/ww.plugins/panels/admin/
    jquery.inlinemultiselect.js"></script>
```

And now, we can amend the `admin/js.js` file to use it. Add these highlighted lines to the end of the `$.get()` section in `panel_visibility()`:

```
    });
    $('select').inlinemultiselect({
      'separator':', ',
      'endSeparator':' and '
    });
  });
}
```

And now, the page selection is much more friendly:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://cms/ww.admin/plugin.php?_plugin=panels&_page
Kae's home page
http://cms/w...&_page=index
WebME
Pages
Site Options
Panels
Widgets
Drag a widget into a panel on the
content-snippet
Add small static HTML snippets to any panel - address, slogan, footer, image, etc.
This panel will be visible in home and home/test [Change...]. If you w be visible in all pages, please ch to indicate that no filtering shoul place.
none
all
reset
close
contact-us
home
home/test
home/test3
home2
image-gallery
second-page
Close
Save
Done
YSlow

You can see what's happened. When the dialog opens, the list of selected pages is shown inline in the text (you can see the words **home and home/test** are bold. When **[Change...]** is clicked, a pop up appears with the list of pages shown in a check-box version of a multi-select).

Behind the scenes, the original clumsy multi-select box has been removed and converted to this nice check-box version.

When submitted, the check-boxes act exactly the same as the multi-select box, so the server can't tell the difference.

Now, write the save file, `/ww.plugins/panels/admin/save-visibility.php`:

```
<?php
require $_SERVER['DOCUMENT_ROOT'].'/ww.admin/admin_libs.php';
if(isset($_REQUEST['id']) && isset($_REQUEST['pages'])){
  $id=(int)$_REQUEST['id'];
  $json='['.addslashes($_REQUEST['pages']).']';
  dbQuery("update panels set visibility='$json'
      where id=$id");
}
```

Again, the server-side code is very simple.

We check that the submitter is an admin, then record what was submitted directly into the panels table.

Panel page visibility—front-end code

The front-end code is very simple.

Edit the `/ww.plugins/panels/plugin.php` file and replace the "`is the panel visible?`" comment block with this code:

```
// { is the panel visible?
if($p['disabled'])return '';
if($p['visibility'] && $p['visibility']!='[]'){
  $visibility=json_decode($p['visibility']);
  if(!in_array($GLOBALS['PAGEDATA']->id,$visibility))
    return '';
}
// }
```

The visibility field in the table is an array of page IDs. If there are any IDs in the array and the ID of the current page is not one of them, then the panel is returned blank.

Widget page visibility

The final piece of the Panels plugin is how to manage widget visibility.

Similar to panels, widgets are not always necessary on every page.

For example, you may have a widget which displays the contents of an online store basket. This widget should not be shown on a page where the online store's checkout shows the same list.

Or maybe you have some widgets that you want to only appear on very specific pages, such as showing an RSS feed from the local cinema on a page which reviews a film.

The script works the same way as the panel visibility code.

Open `/ww.plugins/panels/admin/js.js` and replace the `widget_visibility()` function stub with this:

```
function widget_visibility(ev){
  var el=ev.target,vis=[];
  var w=$(el).closest('.widget-wrapper');
  var wd=w.data('widget');
```

```
      if(wd.visibility)vis=wd.visibility;
      $.get('/ww.plugins/panels/admin/get-visibility.php?'
        +'visibility='+vis,function(options){
        var d=$('<form><p>This panel will be visible in <select '
          +'name="panel_visibility_pages[]" multiple="multiple">'
          +options+'</select>. If you want it to be visible in '
          +'all pages, please choose <b>none</b> to indicate '
          +'that no filtering should take place.</p></form>');
        d.dialog({
          width:300,
          height:400,
          close:function(){
            $('#panel_visibility_pages').remove();
            d.remove();
          },
          buttons:{
            'Save':function(){
              var arr=[];
              $('input[name="panel_visibility_pages[]"]:checked')
                .each(function(){
                  arr.push(this.value);
                });
              wd.visibility=arr;
              w.data('widget',wd);
              updateWidgets(w.closest('.panel-wrapper'));
              d.dialog('close');
            },
            'Close':function(){
              d.dialog('close');
            }
          }
        });
        $('select').inlinemultiselect({
          'separator':', ',
          'endSeparator':' and '
        });
      });
    }
```

You can see that this is very similar to the panel visibility code. The main difference is that the panel code calls the server directly in order to save the page IDs, while this code records the page IDs in the widget data contained in the panel and then calls `updateWidgets()` to record it.

On the front-end, the code is just as simple as the panel code. Add the highlighted lines to `/ww.plugins/panels/plugin.php`:

```
if(isset($widget->disabled) && $widget->disabled)continue;
if(isset($widget->visibility)
    && count($widget->visibility)){
  if(!in_array($GLOBALS['PAGEDATA']->id,
      $widget->visibility))continue;
}
if(isset($widget->header_visibility)
    && $widget->header_visibility)
  $h.='<h4 class="panel-widget-header '
  .preg_replace('/[^a-z0-9A-Z\-]/','',$widget->name)
  .'">'.htmlspecialchars($widget->name).'</h4>';
```

It's the same idea as with panels—we check to see if a list of page IDs is recorded. If there is and the current page is not in the list, then we don't go any further in the loop with this widget.

Summary

In this chapter, we enhanced and completed the panels and widgets system such that you could disable them, choose which pages they were visible on, and customize the widgets.

In the final chapter, we will build an installer for the CMS.