Template Documentation for AxDCMS 0.1.1

It is recommended you disable check spelling & grammar as you type while reading this.

Contents

Forward

Well let's see. The templating system runs on Smarty, and this will continue to be in future versions (no guarantees, just my current stance at the moment).

This documentation will help you make your own theme. It's not that complex; however keep in mind this templating system will be deprecated in 0.1+, mainly because the current one is not very good for either module writers or template writers, so I'm taking the idea behind Post-Nuke's and making it actually work. The theme you make however should be able to be ported *fairly* easily.

Templates are stored in /templates/TEMPLATE_NAME/, where TEMPLATE_NAME is the name of your template. There also must be a TEMPLATE_NAME.conf file. Each template must be named with a .tpl, and they are listed below. Javascript(with modifications), CSS (but no <style></style> in a .tpl, linked only), XHTML or HTML, DHTML, Flash, and Images may all be used in the making of the template.

At the very least templates for 0.1.1+ will need modification, so if you write one, try to keep up to date. This document will be kept up to date. Let's get started.

Syntax

Syntax of the templating system is what smarty uses. If you are unfamiliar, here's a quick rundown. You should also read the entire document to get a better understanding.

Variables

Variables are declared within the program. You use them in smarty as {$VAR_NAME}, replacing VAR_NAME with the name of the variable.

Sections

Sections are used in the template when there are sets of variables that need to formatted the same way, and it isn't known what these are or how many there are before runtime. A sample would be when you have a series of links, such as navigation. For simplicities sake, we are going to just list them.

```
{section name=lid loop=$links}
&middot;<a href="{$links[lid].href}">{$links[lid].text}</a>&middot;
{/section}
```

Let's disect this. {section name=LOOP_N loop=$LOOP_N}

LOOP_N is used by smarty to decide which item it is using. All that this does is make it so that whatever LOOP_N is, it's what goes in the [] of the variable.

LOOP_NAME is the name of the list. A section loop is basically a big numbered list, each list having items. These items are what you would use inside the {section …} {/section} These will be listed in the section describing the template.

The next part is inside the section. In order to access a value from a section, we must call it differently from a standard smarty variable. The syntax is {$LOOP_NAME[LOOP_N].ITEM_NAME}. ITEM_NAME is the name of the item you are trying to access. These will be listed with each LOOP_NAME.

The last part is the closing tag. This goes after all the html/xhtml code you wish to repeat. It is and always will be {/section}

Look at our code sample again. As you see, the name of the loop is links. The "LOOP_N" is lid, which is short for **l**oop **id**. It is good programming practice to name things in a uniform way. There are two items in our loop, href and text. If you understood the above, the code will generate the following:

·Link 1·Link 2·Link 3·

Assuming the three item "text"s are Link 1, Link 2, Link 3, all with "href"s of #. If there were more, they would automatically be tacked on. That's the beauty of sections. Config file syntax is covered on page 17.

error.tpl

This is displayed when AxDCMS needs a simple box, like a "Successfully logged in." or "News Item Does Not Exist." There are two variables in this.

title
This variable stores the title of the message. This should probably put in a high priority place. Let's look at the image from my theme. In this case, "User Saved" is the title. It is at the top of the box, clear, strong, and important.

message
This variable is the content. In the case of my template, it is the message "The user… settings.

It's simple enough. The error template does become the body variable the main template.

login.tpl

In this template, we are introduced to l_, u_, f_, and v_ prefixes that are standard in AxDCMS themes.

l_login
This variable is the language for User Name. l_ stands for language, and is used to allow for multiple languages in AxDCMS. In this case, it is used as both the title of the content box, and the button. Below, I am just going to list the language variable and what it refers to on the screen shot.

l_username "User Name", colon is not included, you must provide it (or not) yourself.
l_password "Password", once again no colon.
l_remember "Remember Me", no colon.
l_register "Register" **l_yes** "Yes" **l_no** "No"
errors "Invalid Username or Password." It will not be empty unless the user supplied an invalid username or password.

Because login.tpl is a form, form fields must be used. The <form> tag is used. The method(method="") is "post" in this form. This needs to go around all the form elements. In the form tag action should be

```
action= "{$form_action}"
```

There will be sample code provied. You may notice that there are no options like class or style. These are allowed, and so are all other options, but when using id(id=""), make sure it has the same content as name(name="").

The first form element is for the username. It is set up like this:

```
<input name="{$f_username}" value="{$v_username}">
```

f_username is the name of the username field. This needs to be smarty'd incase it changes in a later version. **v_username** is the value of the field, used if the login is invalid, so the field is automatically filled.

The next form element is the password field. It is similar to above, but it is of type="password", and it has no value.

```
<input type="password" name="{$f_password}" value="">
```

Next we come to the radio buttons. These are both named (name="") {$f_remember} The Yes option has a value (value="") of {$v_remember_true} where as the No option's value is {$v_remember_false}

```
<input
 name="{$f_remember}"
 value="{$_remember_true}" type="radio">{$l_yes}
<input
 checked="checked"
 name="{$f_remember}" value="{$v_remember_false}"
 type="radio">{$l_no}
```

Next, we have a register link. The language var was listed above. In addition to that, we have a u_ variable. u_ is a prefix for url, and will be used most of the time in a link. For this register link, we have **u_register**. In conjunction with the l_register variable, we have this:

```
<a href="{$u_register}">{$l_register}</a>
```

Links are also allowed to have options not listed in the samples.

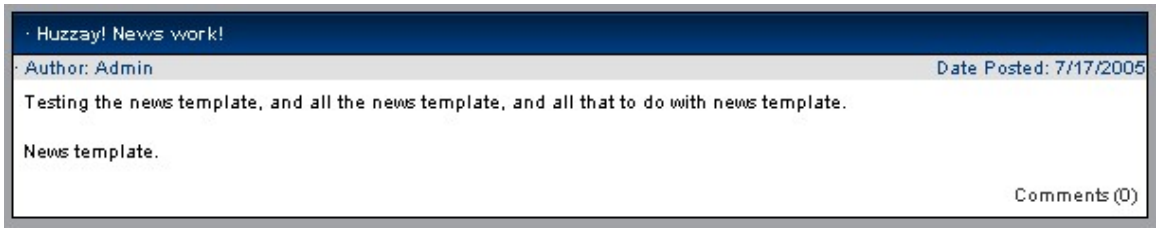
login.tpl also has a hidden field used in redirection. It's name(name="") is {$f_location} and it's value(value="") is {$v_location}

```
<input type="hidden" name="{$f_location}" value="{$v_location}">
```

If this is not in there, serious errors can occur.

Finally, we have the submit button. So long as it's value(value="") is {$l_login} you can do whatever you want with it as long as it's there.

news.tpl



title	The title of the news post, in this case, "Huzzay! News work!"
l_author	"Author" variable (no colon included, add it if you want)
author	The author of the news article, in this case, "Admin"
l_date	"Date Posted" variable.
date	The date the article was posted, "7/17/2005" in this case.
body	The content of the news post, "Testing…template." in this case.
u_comments	The url of the comments page, usually used in a <a href="">
l_comments	"Comments" variable.

n_comments
The number of comments, "3" in this case. This does not include the ( ) for the reason that it should be up to the template designer. He/She/It can use [ ], or whatever, just make sure that if you use { and }, they are written as
{literal}{{/literal}{$n_comments}{literal}}{/literal}
If you want, you can even switch the order so it looks like "3 Comments", however "1 Comments" might look funny, but research smarty and you'll find a fix.

{literal}…{/literal} are used when you want whatever is between them to be ignored by smarty. This can fix problems such as when an error occurs "Unrecognized tag {{$n_comments}}" will occur if you try to use { } around the number of comments. You must wrap each bracket into a {literal}{/literal} set, like
`{literal}{{/literal}{$n_comments}{literal}}{/liternal}.`

This is the same when using with javascript or css, or whatever else uses { and }, smarty delimiters.

news_comments.tpl

title	The title of the news post, in this case, "Huzzay! News work!"
l_author	"Author" variable (no colon included, add it if you want)
author	The author of the news article, in this case, "Admin"
l_date	"Date Posted" variable.
date	The date the article was posted, "7/17/2005" in this case.
body	The content of the news post, "Testing…template." in this case.
l_comments	"Comments" variable.
l_by	"by"
l_add_comment	"Add Comment" variable.
l_title	"Title" variable.
l_message	"Message" variable.
l_add	"Post" variable, as show on the button.

nocomments
This variable is either 1 if there are comments, and 0 if there isn't any. Why is this useful? For one simple reason: the {if} tag. Using the {if} tag, we have more freedom with our template. Let's say we had no {if} tag. We could not have something like this:

```
<table><tr><th colspan="2">{$l_comments}</th></tr>
{section name=cid loop=$comments}
<tr><td bgcolor="#CCCCCC"><span style="FONT-SIZE: 10px;">
{$comments[cid].title} {$l_by} {$comments[cid].author}</span></td><td
align="right"><span style="FONT-SIZE: 10px;">{$l_date}
{$comments[cid].date}</span></td></tr>
<tr><td colspan="2">{$comments[cid].body}</td></tr>
{/section}
</table>
```

If there were no comments, then there would be a bulky empty table with “Comments”, and nothing below it. This is not what we want. So we modify the code to only display the table if there are comments.

```
{if $nocomments ne 1}
<table><tr><th colspan="2">{$l_comments}</th></tr>
{section name=cid loop=$comments}
<tr><td bgcolor="#CCCCCC"><span style="FONT-SIZE: 10px;">
{$comments[cid].title} {$l_by} {$comments[cid].author}</span></td><td
align="right"><span style="FONT-SIZE: 10px;">{$l_date}
{$comments[cid].date}</span></td></tr>
<tr><td colspan="2">{$comments[cid].body}</td></tr>
{/section}
</table>
{/if}
```

If you noticed, we have a new term, “ne”. This is short for “not equal to”. In AxDCMS, if nocomments is equal to 1, then there are no comments (clever huh?). Equal to anything else means there are comments. You don’t have to use this, but when I was making my template, I needed it, so I added it.

comments
Comments is a section. We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won’t list it again. Below, until specified, are only items in the comments section, not stand alone variables.

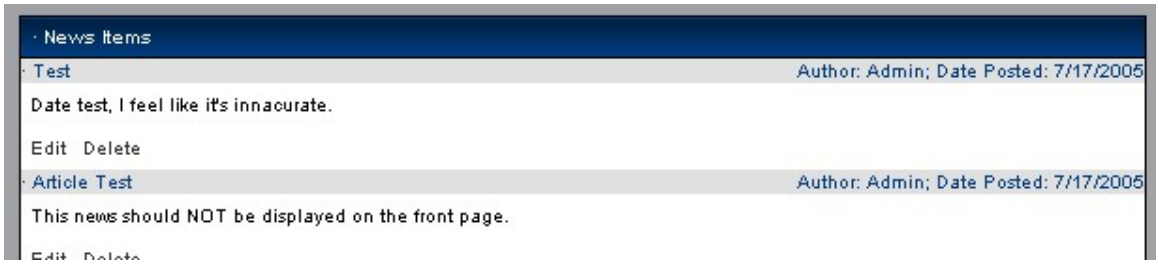
title	The title of the comment. “Test Comment” in the above example.
author	The author of the comment. “Admin” in the above example.
date	The date the comment was posted. “7/17/2005” in the above example.
body	The body of the article. “Your…Gordon.” in the above example. This is indeed a famous quote from my favorite game.

That’s all of them. These must be used inside the section as $comments[LOOP_N].ITEM, where LOOP_N is whatever name is, in the above code, cid. ITEM is the item you are trying to access, in this case, title, author, date, body. See the syntax chapter for more specific items.

You may have also noticed that this page has a form. The values (value=“”) are empty. Submit can be named(name=“”) whatever you want. The field names are as follows:

f_message	The name(name=“”) of the message field (It’s a textarea in the above, but it doesn’t have to be a textarea)
f_title	The name(name=“”) of the Title field.
form_action	The action of the <form …>. Method(method=“”) is “post”

news_admin.tpl



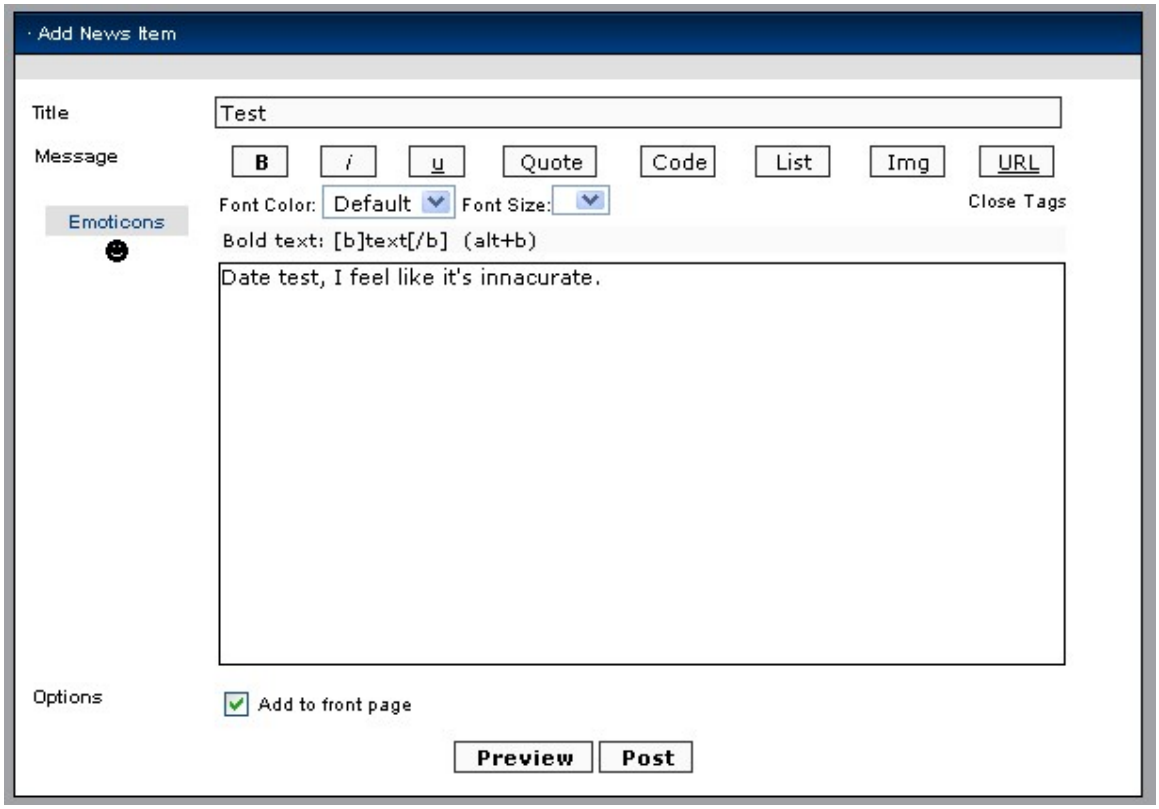
l_news "News Items" variable.
l_by "by" variable.
l_date "Date Posted" variable
l_edit "Edit" variable
l_delete "Delete" variable

news
news is a section. We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won't list it again. Below, until specified, are only items in the comments section, not stand alone variables.

title The title of the post. "Test" and "Article Test" in the above image, second news post.
author The author of the post. "Admin" in both items in the above example.
date The date the news was posted. "7/17/2005" in the above image.
body The body of the article. "Date…innacurate." in the above image, top item. You can set the maximum characters this is before it is trimmed in the config file. If it is trimmed the extention (… for my template) can be set in the config file.

That's all of them. These must be used inside the section as $comments[LOOP_N].ITEM, where LOOP_N is whatever name is. ITEM is the item you are trying to access, in this case, title, author, date, body. See the syntax chapter for more specific items.

news_admin_add.tpl



If you haven't noticed by now, I have taken a lot of tips from other PHP programs, phpBB being one of them.

I'm just going to list most of the variables for my sanity. These are basically a copy->paste find->replace from the source code. Some I've gone into more detail with. Anyhow, on with the documentation.

`smilies`

`smilies is a section.` We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won't list it again. Below, until specified, are only items in the comments section, not stand alone variables.

code	The smilie code. The one smilie that comes with AxDCMS is ":)"
text	The smilie description. For example, "smile"
src	The image name of the smilie. This has everything, just pop it into the src(src="") attribute of an image tag.

That's all of them. These must be used inside the section as $smilies[LOOP_N].ITEM, where LOOP_N is whatever name is. ITEM is the item you are trying to access, in this case: code, text, or src. See the syntax chapter for more specific items.

table_name "Add News Item" variable.

l_title "Title" variable.
l_message "Message" variable.
l_smilies "Emoticons" variable.
l_B "B" variable.
l_i "i" variable.
l_u "u" variable.
l_quote "Quote" variable.
l_code "Code" variable.
l_list "List" variable.
l_img "Img" variable.
l_url "URL" variable.
l_fontcolor "Font Color" variable.
l_default "Default" variable. This should be right before the font_c section, as this is not included in that section.

font_c
font_c is a section. Sections are described in the syntax section, and are used in above templates. font_c is used to loop the option tags in Font Color: dropdown. The following variables are part of this section until otherwise specified.

v This is the lowercase one word version, the html color. e.g darkred or green.
t This is the text of the color option.

That's all of them. To assist understanding, a sample is included below from xDeep.

```
{section name=c loop=$font_c}
   <option style="color:{$font_c[c].v}; background-color: #FAFAFA"
   value="{$font_c[c].v}" class="content_body">{$font_c[c].t}</option>
{/section}
```

l_fontsize "Font Size" variable.

font_s
font_s is a section. Sections are described in the syntax section, and are used in above templates. font_s is used to loop the option tags in Font Size: dropdown. The following variables are part of this section until otherwise specified.

v This is the size, in px, of the font.
t This is the text of the size option.

That's all of them. To assist understanding, a sample is included below. This is from the xDeep template.

```
{section name=s loop=$font_s}
<option value="{$font_s[s].v}"
class="content_body">{$font_s[s].t}</option>                    {/section}
```

l_closetags "Close Tags" variable.

l_options	"Options" variable
l_mustMessage	"You must enter a message when posting." variable.
l_mustTitle	"You must enter a title when posting." variable.
v_helpline	"Tip: Styles can be applied quickly to selected text." variable.

f_frontpage, **v_frontpage**, **c_frontpage**, **l_frontpage**
These make up the checkbox & "Add to front page" part of the form. An exerpt from the default template "xDeep" is included below.

```
<td><input type="checkbox" name="{$f_frontpage}" value="{$v_frontpage}"
{$c_frontpage}  /></td>
<td><span class="content_body">{$l_frontpage}</span></td>
```

l_preview
This field is important. This MUST be the value(value="") of the preview button, or it will not work. I don't know how to remove this dependency, so until someone better with php and forms does, stay with me on this.

l_submit
Important like the above, only it has to be the value(value="") of the Post button.

f_message
The name of the Message text area. It doesn't have to be a text area, but it has to be named message. This is for the field that the body of the news item will be entered into.

v_message
This is the value of the message field, used in previewing and editing.

f_title
The name of the title field, in the picture, it's the one to the right of "Title"

v_title
This is the value of the title field, used in previewing and editing.

f_submit
This one is important. It is the name(name="") value for both the Preview button and the Post button.

f_poster
This template has one hidden field for it's form, the poster field. f_poster is the name(name="") attribute, **v_poster** is the value(value="") attribute.

v_errors
A list of errors. This is shown as bright red text next to "Add News Item", and it fits on one line to give you a picture of size. I don't suggest the bright red text, unless it matches. You can put this where ever you think it looks best.

form_action
This is the action(action="") of the <form …> tag in your template. The method(method="") is "post".

The above is complicated, long, and was a real pain to make, so below, I have a list of variables you will need if you want make it simple, e.g. a title box, a message box, a preview button, and a submit button.

l_title	l_message	l_preview
l_submit	f_message	v_message
f_title	v_title	f_submit
f_poster	v_poster	v_errors
form_action	table_name	

Barebones will make it much easier. 0.0-2 will make it much, much easier, for the if the script is included, it will be placed into the language dir, and the link code will be called via a {$javascript_link} or what not. I may also put things like the font drop downs into a section to make it easier. And of coarse, the "Options" row will reappear with an "Is Article" option, making it so that it doesn't appear on the front page.

user_regedit.tpl

If you notice I am getting vaguer and vaguer as this gets longer and longer. If you were clever, you wouldn't wonder this because you would see that the images are taller than the one's at the beginning. That and you should know this by now. Anyhow.

e_show
Remember the {if} from a while back? It's used here again. The following is an example.

```
{if $e_show eq 1}
<table width="549px" cellspacing="1" cellpadding="6" border="0">
  <tr>
      <td class="error">
        {$e_body}
      </td>
  </tr>
</table><br />
{/if}
```

Let's look at the code. "eq" is much like "ne", only "eq" stands for "equal to", the opposite of "ne" So if e_show is equal to 1 we'll show the error table. If not, do nothing.

e_body
The errors generated by a bad registration. These can add up and become big, so putting the in the title of the content box is a bad idea. I put them in a nicely padded box up top. You can do what you like. What these are separated by is set in the template config file.

l_registrationinfo	"Registration Info" variable.
m_starred	"Items marked…otherwise." variable.
l_username	"Username" variable, no colon or star. The star will be controlled server side after the big template coding change (yes, yes indeed).
l_email	**"**E-mail Address" variable, no colon or star.
l_newpass	"Password" variable, no colon or star.
m_newpass	"The password…log in" variable.
f_newpass	This is the name(name="") attribute of the new password field.
form_action	The action(action="") attribute of the <form …> tag. Form is post.

s_admin
This is an important variable, used in an {if}. If it equals 1, then the template is being called from an admin page. This requires a slight change of fields. If it does equal 1, you want to add a radio option to set the user to admin status or not. This is set to change to a rank drop down in a future version, when ranks are implimented. If it doesn't equal 1, you want a password verification field. Let's look at some sample code.

```
{if $s_admin eq 1}
<tr>
<td>{$l_isadmin}:</td>
<td>
<input type="radio" name="{$f_isadmin}" value="1" {$v_isadmin_y} />
{$l_yes}
<input type="radio" name="{$f_isadmin}" value="0" {$v_isadmin_n} />
{$l_no}
</td>
</tr>
{else}
<tr>
<td>{$l_confpass}: * <br />{$m_confpass}</td>
<td><input type="password" name="{$f_confpass}" value="" /></td>
</tr>
{/if}
```

As you can see, it ended up working out perfectly. In admin, isadmin is at the bottom. In register, confirm password is at the bottom, below new password. Back to the variables.

f_isadmin	This is the name(name="") attribute of the isadmin radio buttons.		
v_isadmin_y	This one is curious. If the database says that the users that is being edited is an admin, this will equal to "checked='checked'", otherwise it will be blank, and **v_isadmin_n** will be equal to "checked='checked'". These aren't required, but sure are helpful.		
l_confpass	"Confirm Password" variable, no colon or star.		
m_confpass	"You…typos" variable.		
f_confpass	The Confirm Password field name(name="")		
l_yes	"Yes" variable.	**l_no**	"No" variable.
l_submit	"Submit" variable	**l_reset**	"Reset" variable.
f_username	This is the name(name="") attribute of the Username field.		
v_username	This is the value(value="") attribute of the Username field.		
f_email	This is the name(name="") attribute of the Email field.		
v_email	This is the value(value="") attribute of the Email field.		

user_admin.tpl

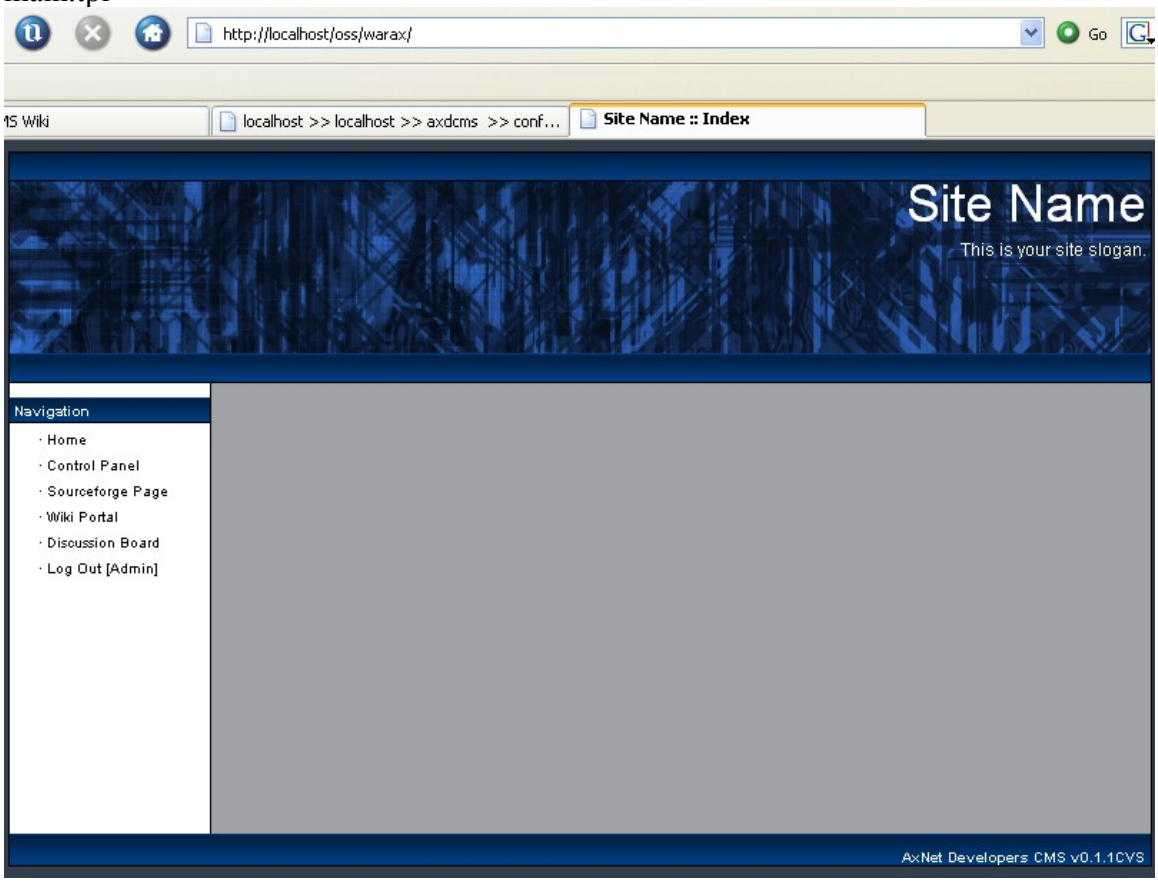
l_delete "Delete" variable.
l_edit "Edit" variable.
l_users "Users" variable.

user
`user is a section.` We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won't list it again. Below, until specified, are only items in the comments section, not stand alone variables.

i This is 1 or 2, useful in alternating row colors as I did.
```
<td class="row{$user[uid].i}"
```
rank This is 1 for admin, 0 for normal. I used an {if} to change the text class for admins to the tealish color.
name The name of the user, eg "T0x1c" or "bisbebri"
email The user's email. I used it in hyperlinks as mailto:{$user[uid].email} combos.
u_edit The url of the user edit form. I used it in an <a> around {$l_edit}
u_delete The url of the user delete form. I used it in an <a> around {$l_delete}

That's all of them. These must be used inside the section as $user[LOOP_N].ITEM, where LOOP_N is whatever name is. ITEM is the item you are trying to access, in this case: code, text, or src. See the syntax chapter for more specific items.

main.tpl



Ah, main.tpl. It's on every page loaded. All the other templates we have covered are just a variable in this, literally.

body This should be placed in the place where you want the other templates to fit into, in my case, the white space below the navigation bar, but above the footer and copyright.

site_title This is the title of the site, what goes between the <title></title> tags in a standard web page.

footer This is the footer of the site. At the moment, it says the app name (AxNet Developers CMS) followed by the version (v0.1.1). The footer is pretty much required, the users of your template won't be too happy when I deny them support because support is only offered to the people who keep the footer. Thanks for the idea phpBB!

nav_main nav_main is a section. The variables are listed below.
href The complete url, should be href="{$nav_main[LOOP_N].href}"
text The text of the link. Example "Home" from my nav bar.

Config File

New in version 0.1.1, template configuration files. These are simple pieces of code that do things that the templates can't do, like change the suffix to a trimmed news post for news_admin, or set the breadcrumb separator (breadcrumbs are on the roadmap for 0.1.2). They also give information about your template, like the name, the author, and a description. These are named TEMPLATE_NAME.conf in your template dir. Let's look at xDeep's template config, /templates/xDeep/xDeep.conf

```
// This is a template config file
// Syntax is simple, variable=value
// This is used to set configuration values for the template

name=xDeep
description=xDeep is the default template, written for AxDCMS 0.1.1.
It is pretty, but is probably over the top and will be changed by 0.2.
author=T0x1c

news_admin_trim=256
news_truncate_tail=...
error_separator=<br />
```

Any line that begins with // is ignored by AxDCMS. You can use as many of these as you want, they will all be ignored. Blank lines are also ignored, as long as they are blank and don't have a space, tab or anything but nothing.

There are six variables in 0.1.1: name, description, author, news_admin_trim, news_truncate_tail, and error_separator.

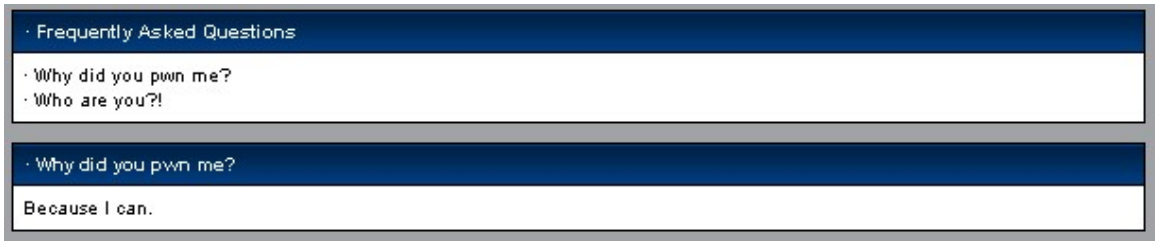
The name of the variable cannot have spaces, and there should be no spaces on either side of the equal sign, however the value of the variable can. These are written VARIABLE_NAME=VARIABLE_VALUE. There can be no newlines (enters, return carriages) Because of this strict syntax, it is recommended that you simply modify xDeep's config file.

name	This is the name of your template. It should be short, but can have spaces.
description	Briefly describe your template here. This should be on one line, to make sure disable word wrap in your editor. Spaces are allowed.
author	This should be your name or username. Spaces are allowed.

There are also some configuration variables.

news_admin_trim	This is the maximum characters to allow the body in news_admin.
news_truncate_tail	This is what appears after the body if it is larger than the news_admin_trim in news_admin.
error_separator	This is what separates multiple registration errors during register.

faq.tpl



l_faq "Frequently Asked Questions" variable.

list
list is a section. We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won't list it again. Below, until specified, are only items in the comments section, not stand alone variables.

href Link for the question, goes in a <a href=""> href(href="") value.
text The text for the question, "Who are you?!" is one example here.

That's all of them. These must be used inside the section as $list[LOOP_N].ITEM, where LOOP_N is whatever name is, in the above code, cid. ITEM is the item you are trying to access, in this case: href and text. See the syntax chapter for more specific items.

qna
qna is a section. We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won't list it again. Below, until specified, are only items in the comments section, not stand alone variables.

aname The name of the anchor that the question is assigned.
question The question, "Why did you pwn me?" in the above example.
answer The answer. "Because I can." in the above example.

That's all of them. These must be used inside the section as $qna[LOOP_N].ITEM, where LOOP_N is whatever name is, in the above code, cid. ITEM is the item you are trying to access, in this case: aname, question, and answer. See the syntax chapter for more specific items. See xDeep/faq.tpl for a complete example.

faq_list.tpl

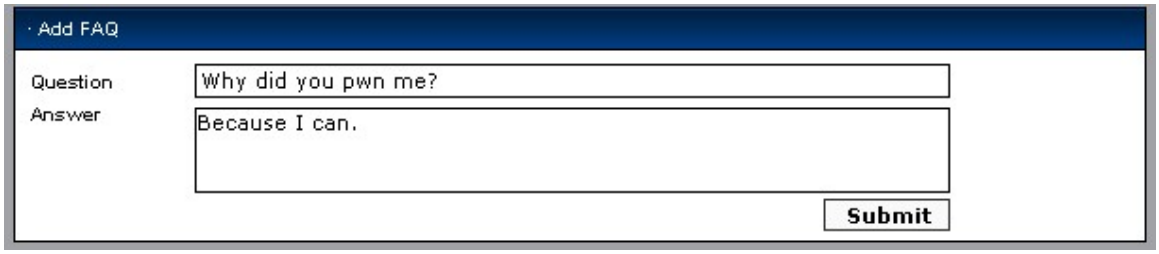
l_faq "Frequently Asked Questions" variable.
l_edit "Edit" variable.
l_delete "Delete" variable.

faq
faq is a section. We discussed this in the syntax chapter. Below are the list of items. The sample is above, listed twice, so I won't list it again. Below, until specified, are only items in the comments section, not stand alone variables.

i This alternates 1 and 2, useful if you want to alternate row styles like me.
question The question, "Why did you pwn me?" is one in the above image.
u_edit The url for editing, this is the href(href="") for the Edit anchor above.
u_delete The url for deleting, this is the href(href="") for the Delete anchor above.

That's all of them. These must be used inside the section as $faq[LOOP_N].ITEM, where LOOP_N is whatever name is, in the above code, cid. ITEM is the item you are trying to access, in this case: aname, question, and answer. See the syntax chapter for more specific items. See xDeep/faq.tpl for a complete example.

faq_form.tpl



l_add_faq "Add FAQ" variable.
l_question "Question" variable.
l_answer "Answer" variable.
l_submit "Submit" variable (the button's value(value="”))

faq_form is, as the name applies, a form. It has no hidden fields.

form_action The action(action="”) of the <form> tag. The method is post.

f_question The name(name="”) of the Question field.
v_question The value(value="”) of the Question field.

f_answer The name(name="”) of the Answer field.
v_answer The value(value="”) of the Answer field. If this is a textarea, like in the above example, it goes between the <textarea …></textarea> tags.

That's it, you have now made an AxDCMS 0.1.1 template. Stay updated as the variables and pages will most likely change in upcoming 0.1.* versions.