Hiding Private Content in CMS Made Simple

How to limit content and pages to only certain authorized users

Author: Robert Campbell

Created: April 13, 2009

Revised By: Robert Campbell

Last Revised: August 15, 2009

Table of Contents

LICENSE:

Introduction:

Many websites, beside publishing information and functionality for the general anonymous visitor, publish information and/or provide functionality that is not intended for the general public, but for authorized users only. This requires that visitors to the website be able to perform some type of authentication, and that this authentication can then be used to test whether or not that visitor is permitted access to that content or functionality. This document will describe the steps needed to do that within CMS Made Simple.

CMS Made Simple is an award winning, open source website content management system written in PHP that extensively utilizes a relational database for storing content, settings, and layout information, and uses smarty to dynamically laying out content.

CMS Made Simple (in versions up to and including 1.6.4) provides a mechanism for website administrators to login from different locations and manage the content and layout of the website. However it does not include any mechanism for authenticating visitors to the website. This is the responsibility of third party add-on functionality.

This document will introduce the third party add-on modules that are most commonly used within CMS for authenticating visitors to the website, and for managing content for authenticated users. It will describe how to install and configure these modules in a new CMS Made Simple website. It will describe how to create a mechanism where by anonymous users can register themselves for the website, but still cannot see any restricted content until that registered user is then approved by the administrator. It will discuss how to use these modules in conjunction with the capabilities provided by CMS Made Simple to limit content or functionality in portions of pages, and to limit display of entire pages, including the display of private pages in navigation to only authorized users.

This document will be formatted much like a tutorial, with step by step instructions needed to achieve the goals set out above. However, it is intended to be a reference for the website developer to aide in configuring a fully capable CMS Made Simple website for numerous different groups of authenticated visitors, and for the regular anonymous user.

This document will not discuss topics such as installation of CMS Made Simple in your web server environment, it's basic usage, or issues like UNIX permissions, mail settings, the smarty language or other information that should be understood by the website developer. This document is aimed at experienced website developers who have a familiarity with concepts such as XHTML, CSS, and in developing professional websites using content management systems. The reader should already have a basic understanding of how to work in the CMS Made Simple environment.

This document is also not intended to be complete documentation for each feature of the modules described. It gives a brief overview of the some of capabilities, and some instructions as to how to configure each module, and how to use it on the front end of your website, but does not provide in-depth explanation of every field, template, preference, or option.

Frequently in CMS Made Simple it is possible to accomplish the same goal via many means. This document will describe some of those means, and not every particular method . It also does not included advanced topics like SEO optimization related to restricted pages, the Search module, or restricting access to certain parts of the output of some third party modules.

Document Conventions

This document has attempted to standardize on the appearance of it's various elements to aide in finding things, to break up some monotonous text, and so that you can more easily understand what is happening.

- Code Snippts

```
Code snippets are displayed like this.

Often the lines that need to be changed in each code snippet will be hi-lited in red.
```

- Important Notes

> Important notes will be in red text surrounded by a red shadowed box,

- Steps you should follow

  When there are steps that the user should follow they will be displayed in bullet format with yellow stars. As much detail will be given as possible:

  Step 1

  Step 2

  Step 3

- Desired output

> When desired output is to be illustrated, it wil be displayed in green text surrounded by a green shadowed box.

Additionally, this document assumes progressive knowledge. That means that as you progress through the document, it is assumed that you have learned and understood the steps that were described in detail previously. These steps, if repeating them is required may not be described in as much detail as the document progresses.

Introducing FrontEndUsers

The FrontEndUsers module is a third party add-on module to CMS Made Simple. Written by Robert Campbell (calguy1000@cmsmadesimple.org), it provides the following functionality (in brief):

- A login form
- Logout and Change Settings Capabilities
- "Remember Me" Capabilities
- Complete "User Attribute"[1] Management.
- Lost Password functionality
- Lost Username functionality
- Membership in multiple groups
- Complete Administrative control
- Complete logging capabilities.
- Template Driven
- Complete development API
- (optional) support for Captcha
- More....

The FrontEndUsers module uses the concept of 'Properties', and associates those properties to user groups. Properties can be things like 'Full Name', 'Age', 'City', 'Profile Picture', etc. Properties, when associated with user groups can be optional, required, or hidden. When a property is 'Required', the user is required to maintain a value for that property at all times. Optional properties allow a property value to be empty. Hidden properties allow administrators to associate data with an individual user without the user seeing that data. Administrators can also control the order in which properties are displayed to the user. Administrators must create at least one property, and associate it with at least one group.

The module also provides advanced management of users in its admin panel (located under "Users & Groups >> Frontend User Management" in the CMS Made Simple admin panel.

The 'Forgot Password' functionality of this module relies on the email capabilities of CMSMailer to send out an email containing an authentication code, and a link to a form whereby the authenticated user can reset a forgotten password. This module stores passwords using a one way encryption mechanism, therefore sending the user an existing

1 These are referred to as "Properties" in the FrontEndUsers module.

password, or even exporting the password to a different system is not possible.

This module does not provide the capabilities to control what content is displayed to anonymous, or known users of certain groups. That is the domain of the CustomContent module.

Introducing CustomContent

CustomContent is a small module written by Robert Campbell (calguy1000@cmsmadesimple.org) that uses the information provided by the FrontEndUsers module and interfaces with the smarty templating engine to allow the CMS Made Simple website developer to edit various templates to either display certain information about the logged in user, or to perform various logical tests on that information to control the display of information. This module has no admin interface of its own.

Introducing SelfRegistration

The SelfRegistration module works in conjunction with the FrontEndUsers module to provide functionality to website developers so that anonymous site visitors can register with the website to gain access to content or functionality that is not available to anonymous visitors.

This module does not contain any permanent data. It simply holds the user information temporarily whilst the visitor is in the process of registering with the website. After the visitor has completed the registration process his information is transferred to the FrontEndUsers module and deleted from within SelfRegistration.

When the registration process is complete, and the user information is migrated into the FrontEndUsers module the user will automatically be a member of a group specified in the initial call to the SelfRegistration module. This will be explained below.

SelfRegistration provides limited administration capabilities. Other than configuring the display and email templates, the administrator can only view information for up to 250 users that have not completed the registration process (in practical use, this should never happen). It also provides the ability for the administrator to clean out data from users that have not completed the registration process.

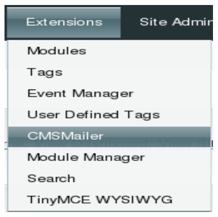
The registration process for website visitors retrieves the list of properties to ask the user from the FrontEndUsers module. Then (in its default configuration) it sends a validation email to the user. Therefore, the FrontEndUsers module must configured to store at least one email address for each user that will be registering via the SelfRegistration Process. Inside the validation email is a link that contains a validation code and the temporary users ID. The user must enter the proper validation code, and the proper password to complete the registration process.

Before You Start

Before you continue with this tutorial, you should have a fresh install of CMS Made Simple 1.5.4 that is working properly. You should have complete administrative access to that install.

Initial Setup

Setup CMSMailer



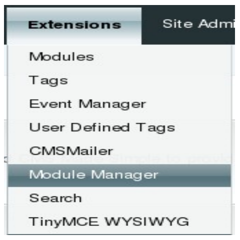
The Selfregistration module (in its default configuration) sends emails to potential registrants so that website administrators can be reasonably assured that the person registering is not a bot, and has a valid email address. Additionally, the FrontEndUsers module uses CMSMailer for its "Forgotten Password" functionality. CMSMailer[2] is the "core module"[3] that provides the functionality for any other module to be able to conveniently and easily send email messages. CMSMailer must be properly configured according to your web hosting environment in order for these modules to function properly.

Your web hosting provider should have provided you with information as to the proper settings for email. These settings may include the SMTP hostname, port number, and a username and password for using email services. You should have these handy when attempting to configure CMSMailer.

Configuring CMSMailer consists of adjusting the settings in the administration panel, and pressing 'submit'... and then entering a valid email address for testing, and pressing the 'Send Test Message' button until such time as a test message is successfully received in your email inbox.

CMSMailer provides numerous options for how emails can be sent. In many UNIX based web hosting environments, the built in php 'mail' may be configured, and you may have success with just adjusting that. If not, you may have to consult your web hosts technical support department for assistance in obtaining the proper settings.

Install the Modules

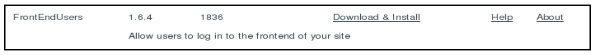


The first step in the process is the installation of the modules. Though there are numerous ways in which this could be accomplished, the simplest way is to use the Module Manager. This "core module" provides the capability to download and expand selected modules directly to the appropriate location in your CMS

2 You can find this module under "Extensions >> CMSMailer" in the CMS Made Simple administration console
3 A module provided with the CMS Made Simple download package and installation.

Made Simple installation.

The Module Manager Screen provides an alphabetical list of the modules that are available for download. It also displays the version number, and a link whereby you can display help and history information on each module without having to install the module. You should browse to the appropriate letter, and then scroll to find the appropriate module, and click the "Download & Install" link.



FrontEndUsers	1.6.4	1836	Download & Install	Help	About
	Allow users to log in to the frontend of your site				

The modules you need to install, are in order:

1. FrontEndUsers version 1.6.4 (or greater)
2. CustomContent version 1.5.3 (or greater)[4]
3. SelfRegistration version 1.2.3 (or greater)

Caution must be used when downloading modules via the ModuleManager. It does not at this time properly check for module dependencies... Therefore, you should check out the history and dependency information in the about link of each module, and download and install the dependencies before downloading the desired module itself.

4 Some issues were detected with the CustomContent module during the writing of this document, and version 1.6 was released in conjunction with the release of this document.

Configuring FrontEndUsers

Once the FrontEndUsers module is installed, you can find it's administration panel in the CMS Made Simple admin console under the "Users & Groups" menu (you may have to refresh the page after the installation process, for the menu to regenerate).

Specifying Module Behavior

You can specify most of the options that change the modules behavior in the "Preferences" tab of the FrontEndUsers administration panel. By default these preferences are fine for most users, however I will briefly explain the important preferences here.

1. "Email Address Is Username"

   This setting indicates to the module that when a users email address is requested by the API, that the username property should be returned. If this option is unchecked, the user will not be able to specify an @ symbol in the username, and a separate email address property will need to be created, marked as required and associated with the appropriate user group(s).

2. "Allow Duplicate Emails"

   This setting indicates that users with the same email address (but different usernames) should be allowed in the system. This may be useful for configurations where numerous people who share the same email address (a sales department for example) need separate logins to an Intranet website.

   Do not use this option in conjunction with the "Email Address is Username" option.

3. "Allow Users to Login More Than Once"

   This setting indicates that the same user may log into the website from two machines (or two browsers on the same machine) at the same time. This option could be used for shared accounts, etc.

4. "Require Membership in at least one Group"

   This setting is an aide to creating and managing users. It enforces that each user record must be associated with at least one member group.

5. "Default Group for New Users"

   This preference indicates which user group should be checked by default in the administration panel, when manually creating a new user account.

   In an initial installation this field will have no valid options in it until at least one user group is properly configured.

6. "Default User Expiry Period"

   Each user account in the FrontEndUsers module has an expiry date. After that expiry date, this user will not be permitted to login to the system. This preference indicates the default period (in months) that will be used when calculating a new users expiry date.

7. "Session Timeout (seconds)"

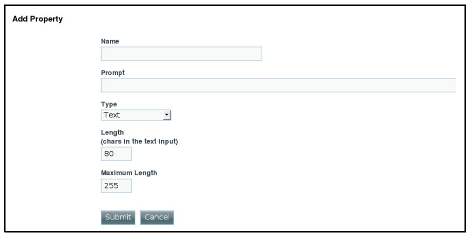
   The FrontEndUsers module will automatically logout users after a certain period of inactivity. This setting indicates how many seconds of inactivity is required.

   Activity is checked for all users each time the FrontEndUsers module is called in one way or another. For example, if one user views the login page, all users that are eligible to be logged out will be removed from the system.

   > The web application has no means to know when a user closes his browser window or navigates away from a FrontEndUsers enabled website. Therefore, if the user closes his browser window, and re-starts it and visits your website again, the system may tell him that he is already logged in. In which case he needs to wait for the automatic expiry period to elapse before re-gaining access to the website.

Creating Properties

The next task that must be completed is to create at least one property inside the FrontEndUsers module. This is accomplished by clicking on the "Add Property" link in the "User Properties" tab of the FrontEndUsers admin panel. Once this happens a form like this will appear:


Add Property

Name

Prompt

Type
Text

Length
(chars in the text input)
80

Maximum Length
255

This form allows you to create a new property within the FrontEndUsers module. There are various different types of properties available, including "Text Area, Email Address, Image, Checkbox, and dropdown list" This allows creating extensive user profiles that are easy for your authenticated users to manage.

The name of the field is what is used internally in the system. Users never see this value. It should consist of all printable ASCII characters and include no spaces *(this is not strictly necessary, but makes things easier later in the process when working with smarty templates)*. For example "full_name" would be a valid property name for the users real name.

The "Prompt" field specifies the prompt that the user will see that is associated with that property. For example "Your Name" would be a valid prompt for the "full_name" property.

For certain types of properties you can specify different options. I.e: for dropdown properties you can specify the options that will appear in that dropdown. For text field properties you can specify the length of the field and the maximum number of characters that can be entered.

Lets do that....

Specify "full_name" as the property name

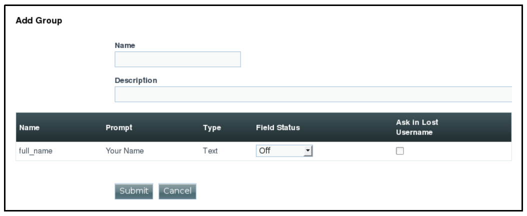
Enter "Your Name" as the prompt

and leave the other settings at their default

click "Submit".

Creating Groups

Now that at least one user property has been created, we can define a user group. This is done by clicking the "Add Group" link on the "Groups" tab of the FrontEndUsers admin panel. You will see a form such as this one:



Add Group

Name

Description

Name	Prompt	Type	Field Status	Ask in Lost Username
full_name	Your Name	Text	Off	☐

Here you can enter a name and a description for the user group that you are creating. The group name should consist of only ASCII characters, and no spaces *(this is not strictly necessary, but makes things easier later in the process when working with smarty templates)*.

As well, you will see a list of the defined properties. Here you can specify how each property is related with that group (Required, Optional, Hidden, or Off), and whether that property should be used in the lost username form for members of that group. If more than one property is defined you will see icons to change the order of the properties.

Create a new group:

Enter "members" for the name

Type "Approved Members" in the description field.

Specify that the "full_name" property is Optional

Click Submit.

Create a second User group

For the purposes of this tutorial, I suggest that you create a second group that is identical to the one created above, except in name. You'll see how we use this second group later.

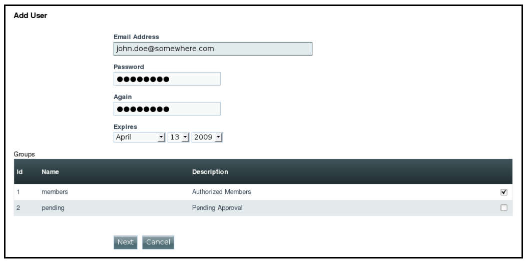
Create a new group:

Enter “pending” for the name

Type “Pending Members” in the description field.

Specify that the “full_name” property is Optional

Click Submit.



Add User

Email Address
john.doe@somewhere.com

Password
●●●●●●●●●

Again
●●●●●●●●●

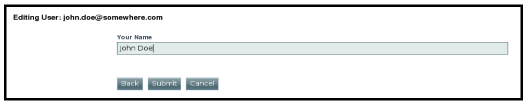
Expires
April 13 2009

Groups

Id	Name	Description	
1	members	Authorized Members	✓
2	pending	Pending Approval	☐

Creating Users

After at least one user property has been created, and at least one user group successfully created, we can manually add users. This is accomplished by clicking the “Add User” link on the “Users” tab of the FrontEndUsers admin panel. Clicking on that link will begin a two step process for the creation of users. As you can see in the image, the first step involves specifying the username, the users password, his expiry date, and which groups that user is a member of. As mentioned above, if a default group is specified in the 'Preferences' tab, it will be selected automatically when this form appears.



Editing User: john.doe@somewhere.com

Your Name
John Doe

The second phase of the user creation process is where the users properties must be defined. As you can see from the snapshot, there is only the one property to be filled out. Had we associated more properties with the 'members' user group there would be more options in this form. Had we specified that one or more of those properties be 'Required' then then those properties would be highlighted in a different color.

Manually Create a User:

Type “john.doe@somewhere.com” for the username

Enter a password for this user

Repeat the new password

Select the “members” group if not already selected

Click “Next”

Enter “John Doe” for the full name

Click Submit

At this stage we are now ready to enable login capabilities for the front end of our website, and to test out those login capabilities.

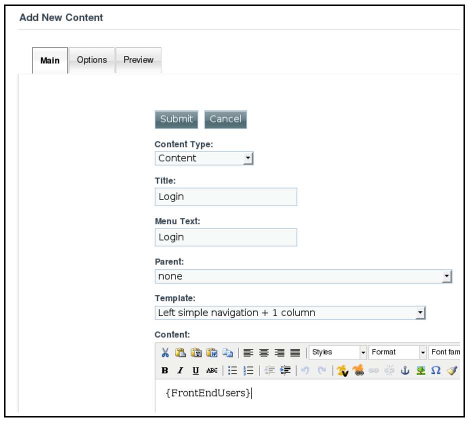
Did you see that we added this user to the 'members' group, and not the 'pending' group? This is because we want this user to be a fully validated user. Later we'll make use of the 'pending' group.

Creating The Login Page

Next we need to allow our test user to be able to login to the website, for him to be able to login, logout, and to change his settings. This is quite trivial as the default behavior of the {FrontEndUsers} tag provides that functionality.

You should go to "Content >> Pages" in the CMS Made Simple admin console, and click on the "Add New Content" link. For a page title specify "Login". For the menu text specify "Login", and for menu text specify "Login", and in the content field, type {FrontEndUsers}. Then click "Submit".

This will create a new page in your CMS Made Simple installation. You can view the front end of your website, and browse to that page to begin testing. You will see the prompt for username, and password, the "Sign In" button, and links that will allow your user to reset his password, or to find a lost username. The layout of this form is controlled by the content of the "Login Template", as found on the appropriate tab of the FrontEndUsers admin panel.



Content Type: Content

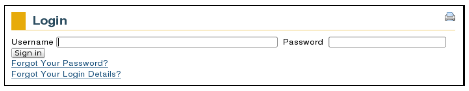
Title: Login

Menu Text: Login

Parent: none

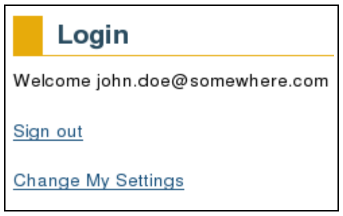
Template: Left simple navigation + 1 column

Content: {FrontEndUsers}



If you enter the username and password that you specified when creating a user above, you will then be presented with an image like the one below. The layout of this content is controlled by the "Logout Template" in the FrontEndUsers admin panel.

This illustrates that the FrontEndUsers module is working, however we have not done anything to our templates to protect any content.



Login

Welcome john.doe@somewhere.com

Sign out

Change My Settings

Configuring SelfRegistration

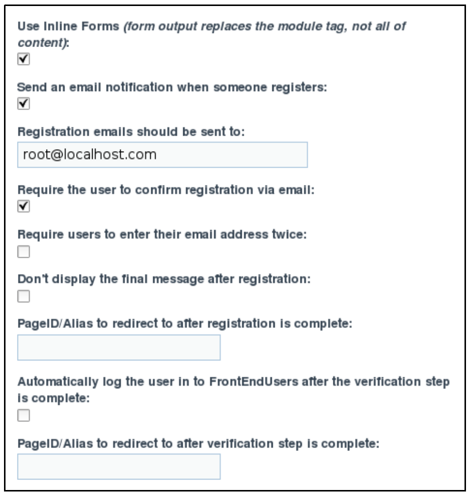
Next we'll configure the SelfRegistration module. There is strictly speaking no need to configure the SelfRegistration module next, however in this case we'll allow users to register themselves first before we restrict the content on this site.

Specifying Module Behavior

As with the FrontEndUsers module, the default options for the SelfRegistration module should be sufficient for the purposes of this document. However I will briefly describe some of the important options in the "Preferences" panel of the SelfRegistration admin panel.

- Use Inline Forms

  This is actually an important concept in CMS Made Simple that is too complex to illustrate in this document, but to summarize, the output of "Inline" forms will replace the original content tag that generated them. The output of non-inline forms will replace content in the {content} tag after the form is submitted. This is important for some layouts, and particularly important with respect to the CustomContent module, as I will illustrate later.



Use Inline Forms *(form output replaces the module tag, not all of content)*:
✓

Send an email notification when someone registers:
✓

Registration emails should be sent to:
root@localhost.com

Require the user to confirm registration via email:
✓

Require users to enter their email address twice:
☐

Don't display the final message after registration:
☐

PageID/Alias to redirect to after registration is complete:

Automatically log the user in to FrontEndUsers after the verification step is complete:
☐

PageID/Alias to redirect to after verification step is complete:

- Send an email notification when someone registers

  This option indicates that an admin person should be notified by email whenever a visitor completes the registration process.

- Registration emails should be sent to

  This specifies the email address that the above notification should go to.

- Require the user to confirm registration via email

  This preference indicates that the user will be required to confirm and validate the email address entered during the registration process. This involves sending the user an introductory email with a link and a randomly generated validation code. This validation code, along with the users username and password must be successfully entered in the validation form in order for the user to complete the process.

  If this option is not checked the user will be instantaneously added to the

FrontEndUsers module without any human or email address verification. It is recommended that this option only be used when Captcha capabilities are provided *(see the Captcha module).*

- Require users to enter their email address twice

  This option indicates that the users, during the registration process must enter their email address twice, and that both entries must match before registration will continue. This is a simple method to avoid typing errors in email addresses during the registration process.

- Don't Display the Final Message after Registration

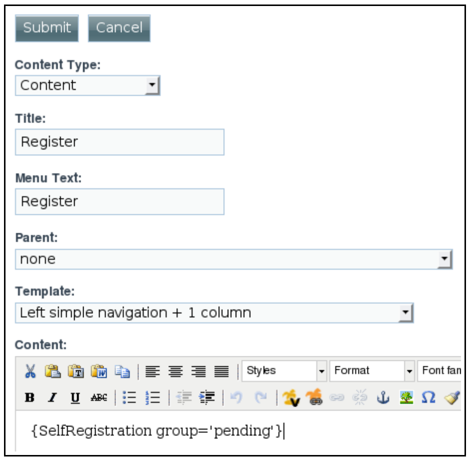
  This option simply states that the 'welcome' message will not be displayed.

- Automatically log the user in to FrontEndUsers after the verification step is complete:

  If this option is selected, after each successful user registration, and the user information is migrated into the FrontEndUsers module, it will instantaneously log the user in. This is useful in conjunction with the "Don't Display..." option above, and Captcha capabilities to allow instantaneous registration and login.

Creating the Registration Page

Adding registration capabilities to a CMS Made Simple website, when FrontEndUsers is already installed and configured is quite trivial. To illustrate this we will create another content page in CMS Made Simples "Content >> Pages" screen. It is not strictly necessary to create yet another page, however it is useful to do it this way in in this document for purposes of clarity.



Content Type: Content

Title: Register

Menu Text: Register

Parent: none

Template: Left simple navigation + 1 column

Content:

{SelfRegistration group='pending'}

Navigate to "Content >> Pages" in the CMS Made Simple admin console, and click "Add New Content". Enter "Register" for both the page title, and menu text, and in the content area enter {SelfRegistration group='pending'}.

Did you notice the group parameter to the {SelfRegistration} tag?. This parameter is required when calling the SelfRegistration module, as it indicates what FrontEndUsers user group the user will become a member of once the registration process is completed.

Later we will be modifying the page template and MenuManager templates to display different content to members, we can also use this same method to display different content to members of different groups.

If we wish to have a process whereby an administrator must manually verify the registration before the registered user can view protected content, we can use this mechanism.   When editing the registered user in the FrontEndUsers admin panel, the administrator can merely change the users group membership from 'pending' to 'member' to mark him as a valid user.

Configuring CustomContent

There is actually no configuration to the CustomContent module. There is no admin panel for this module, so strictly speaking there is nothing to do. I added this paragraph because without it.. certainly somebody would ask the question 'And what do I do with CustomContent?'.

We are ready to begin working on showing different content to different users.

Restricting Visibility

Now we are ready to change our design to display different content or functionality to different users, but first we have to decide how we want things to behave... there are a number of options (which can be combined).

1. Prevent anonymous users from accessing restricted pages completely *(redirect them to the login page if they happen to access the URL without being logged in*
2. Prevent anonymous users from seeing content on restricted pages *(display a nice message)*
3. Show some content on the page, but hide restricted content

For the purposes of this document, we'll start with option 1. We are going to stop any non-loggedin member from landing on restricted pages. If they happen to find a URL to a restricted page and navigate there, we will redirect them to the login page. Later we'll illustrate option 2 because there are some important concepts to learn when using that method.

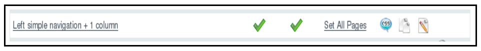
Example 1: Redirection

This example will walk you through a simple method of how to mark certain pages as restricted, how to refuse access to those pages to visitors that have not logged-in to your website, and how to hide those restricted pages from the navigation.

Configuring the Page Template

Here we will introduce the use of CustomContent syntax in your smarty powered page template. We will do a simple test to see if the visitor has authenticated with the FrontEndUsers module, and if not, we will redirect to the login page.

Prior to this though, we'll need to create another page template.... This is accomplished by clicking on the "Copy" icon beside the appropriate template in the "Layout >> Templates" page.



Click on the copy icon beside the template entitled "Left Simple navigation + 1 column".

Call the new template "Left Simple navigation + 1 column restricted"

Click "Submit".

Next, we need to edit the new template:

Click on the "Edit" link for the new template we just created

In the content of that page (you may have to scroll down a ways) you will see this text where the body tag begins.

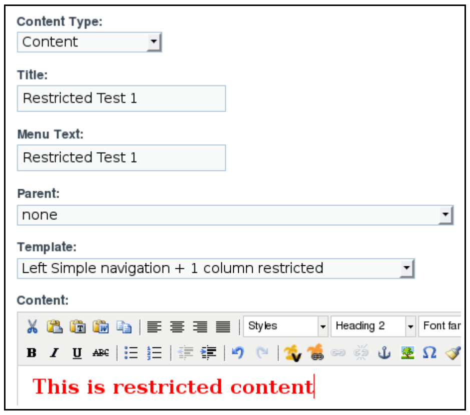
```
<body>

<div id="pagewrapper">
```

Change it to this:

```
<body>
{if !$ccuser->loggedin()}{redirect_page page='login'}{/if}
<div id="pagewrapper">
```

This code introduces the {$ccuser} object that is smarty is aware of, and the {redirect_page} tag that is included with CMS Made Simple. The line we added says: “if the user is not loggedin, redirect to the page with the alias of 'login'. “.

If you weren't aware, CMS Made Simple automatically gave the “Login” page we created above the alias of “login”. Additional documentation and assistance for the {$ccuser} object can be found in the help for the CustomContent module.   You can find this by clicking on the “Help” link in the CustomContent row of “Extensions >> Modules”. Help for the {redirect_page} tag can be found (along with help for all of the other tags included with CMS Made Simple) under the “Extensions >> Tags” menu option in the CMS Made Simple admin console.



Content Type: Content

Title: Restricted Test 1

Menu Text: Restricted Test 1

Parent: none

Template: Left Simple navigation + 1 column restricted

Content:

This is restricted content

Next we need to create a new page that uses that template.

Click on “Add New Content” in the “Content >> Pages” page again. And enter the following information:

Page Title: “Restricted Page”

Menu Text: “Restricted Page”

Template: “Left Simple Navigation + 1 Column restricted” *(the template we just created and edited)*

Enter some text into the content area

Switch to the “Options” tab, and make sure that the “Cachable” option is OFF

When considering this page, it is important to consider that every request for this page could be coming from a different user. Some users may be authenticated, and others may not be. Therefore there is really no way that the content for this page can be cached, it needs to be generated completely on each request so that the appropriate tests can take place.

Every page that has this requirement cannot be cached. You will need to ensure that the cachable option is off for each page that uses CustomContent logic... Including the page where your login/logout forms are located.

Click Submit

We have now created a new content page, using a new page template, that will only allow people that have been authenticated with the FrontEndUsers module, and are currently active to view it.

Testing

Testing the configuration is a matter of clicking on the "Restricted Test 1" link when both logged in, and logged out. When logged out you should automatically be redirected to the "login" page after clicking on the "Restricted Test 1" page.

Follow this procedure:

Click on the "Login" link in the navigation. If you are currently logged in, click on the "Logout" link.

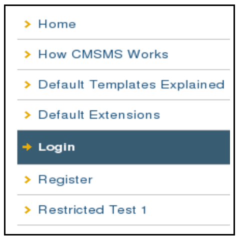
Click on the "Restricted Test 1" link in the navigation

You should be instantaneously redirected back to the "Login" page.

Enter your username and password and click "Sign In"

Click on the "Restricted Test 1" link in the navigation

You should see the text "This is Restricted content"



However, there is a slight issue. If you look at the navigation you will see something like the attached image. The "Restricted Test 1" link shows up in the navigation at all times, whether we are logged in or not.

The next step will illustrate how to customize the MenuManager to solve that problem.

Modifying the Navigation

Modifying the navigation to work the way we want involves:

1. Create a new MenuManager template

    Go to "Layout >> MenuManager", select the "File Templates" tab.

    Click on the "Import" icon beside "simple_navigation.tpl"

    Enter "simple-restricted" for the new template name

Click Submit.

2. Modify our existing page templates to use the new MenuManager template to build the navigation.

   Go to “Layout >> Templates”

   Click on the “Left Simple Navigation + 1 Column” template link

   In the template text area find the line that states

   ```
   {menu template='simple_navigation.tpl' collapse='1'}
   ```

   change it to

   ```
   {menu template='simple-restricted' collapse='1'}
   ```

   Click Submit

   Repeat the above steps on the template named “Left Simple Navigation + 1 column restricted”

3. Set a flag with each restricted page to indicate to the MenuManager that some pages are restricted.

   Go to “Content >> Pages”

   Click on the page entitled “Restricted Test 1” to bring up the edit form

   Navigate to the Options Tab

   Change the “extra1” field to “restricted”

   There are numerous properties associated with each content page. In this document we will use the “extra1” property to flag whether a page is 'restricted' or not.   Strictly speaking we could use any of these properties, or even create our own content block to contain the flag.

   Click Submit.

4. Build tests into the new MenuManager template to test for logged in status, and restricted page status.

   The MenuManager module reads the content hierarchy as defined in “Content >> Pages” and uses that information, a few parameters, and the specified template to build the navigation.   We need to modify template just created to add logic that indicates that the MenuManager should ignore 'restricted' pages, identified by the word 'restricted' in the extra1 attribute of applicable pages.

   Below you will find the appropriate portion of the simple-restricted menu manager template.   The lines I have added are in red. You will see that a simple expression that again tests the {$ccuser} object, and the extra1 attribute on each potential navigation node will result in restricted nodes being ignored if the current visitor is not logged in.

```
{if ($node->extra1 == 'restricted' and $ccuser->loggedin()) or $node->extra1== '' }
{if $node->current == true}
<li class="currentpage"><h3><dfn>Current page is {$node->hierarchy}: </dfn>{$node->menutext}</h3>


{elseif $node->parent == true}
<li class="activeparent"><a class="activeparent" href="{$node->url}"{if $node->accesskey != ''} accesskey="{$node->accesskey}"{/if}{if $node->tabindex != ''} tabindex="{$node->tabindex}"{/if}{if $node->titleattribute != ''} title="{$node->titleattribute}"{/if}><dfn>{$node->hierarchy}: </dfn>{$node->menutext}</a>


{elseif $node->type == 'sectionheader'}
<li class="sectionheader">{$node->menutext}


{elseif $node->type == 'separator'}
<li class="separator" style="list-style-type: none;"> <hr />


{else}
<li><a href="{$node->url}"{if $node->accesskey != ''} accesskey="{$node->accesskey}"{/if}{if $node->tabindex != ''} tabindex="{$node->tabindex}"{/if}{if $node->titleattribute != ''} title="{$node->titleattribute}"{/if}{if $node->target != ''} target="{$node->target}"{/if}><dfn>{$node->hierarchy}: </dfn>{$node->menutext}</a>


{/if}
{/if}{* node->extra1 *}
```

Click on "Layout >> MenuManager" in the CMS Made Simple admin panel

Click on the "simple-restricted" menu manager template to bring up the edit form.

Make the changes as specified in red above.

Click Submit.

Repeat Testing

You should now repeat the testing as indicated above. You will see that the 'Restricted Test 1' page will disappear from the navigation when your test user is currently not logged in to your website. Conversely, when you login to the website using the test account, the 'Restricted Test 1' page will appear.

Summary of Example 1

After quickly reviewing the steps we have followed so far in this document, you will see that we have managed to create a new content page that has content on it that we only want logged in users to see, and that we can successfully stop anonymous people from seeing that content by redirecting them to another page whenever they try to access it.

Similarly, you can see that the restricted page no longer shows up in the navigation when the visitor is not logged in to the website. This is nice. But it has a couple of problems:

1. If you've been following this tutorial, you may have realized that users that successfully register themselves to the website are placed in the 'pending' user group. Those users will still be able to login to the system, and see the content. This is because we are not currently testing (in the page template or the menu manager template) for membership in certain groups.
2. We had to create a second page template that was essentially identical to the first page template except for a small change. This is unfortunate because any changes to layout or logic would have to be done in both templates. This is inefficient, and really not necessary.

It's a good thing these difficulties are easy to overcome.

Example 2: Redirection in 1 Template

This example will fix the errors in the last example by testing for membership in specific FrontEndUser groups, and will eliminate the redundancy of the second template.

Modify The Page Template

As you read in the first example, we added the following logic to our “Left Simple Navigation + 1 Column restricted” template:

```
{if ($node->extra1 == 'restricted' and $ccuser->loggedin()) or $node->extra1== '' }
```

This text has the weakness that it does not test for membership in specific user groups, specifically the 'members' groups. This means that users that have registered themselves with your website will currently be able to see restricted content without you having moved them into the 'members' user group.

Fortunately this is simple to solve. Replace the text above in that template with:

```
{if !$ccuser->memberof('members')}{redirect_page page='login'}{/if}
```

Modify the MenuManager Template

The simple-restricted MenuManager template that we created above, also needs to be changed in a similar way. You need to replace this content:

```
{if $ccuser->loggedin() or $node->extra1 != 'restricted'}
```

with this content:

```
{if ($node->extra1 == 'restricted' and $ccuser->memberof('members')) or $node->extra1==
'' }
```

Testing

To properly test if this procedure works, you will need to register a user account. You can do this by completing the form on the 'Register' Page, and following the steps illustrated in the email that you will receive.

Once you have completed that, then follow these steps:

Ensure that you are logged out from your website

Ensure that the 'Restricted Test 1' content page does not show up in the navigation

Login as your new test user

Ensure that the 'Restricted Test 1' content page STILL does not show up in the navigation.

Remove The Redundancy

Next, we need to remove the redundancy. We really don't need two page templates that are exactly identical except for one line. We'll do this by using the {page_attr} plugin that is standard with CMS Made Simple 1.5.4 to retrieve the value of the extra1 attribute of pages.

Follow these steps:

Edit the “Restricted Test 1” Page, and change the template back to “Left Simple Navigation + 1 Column”

Delete the “Left Simple Navigation + 1 Column restricted” template

Edit the “Left Simple Navigation + 1 Column” template, and directly below the <body> tag set the text to:

```
<body>

{page_attr key='extra1' assign='extra1'}

{if !$ccuser->memberof('members') and $extra1 == 'restricted'}

  {redirect_page page='login'}

{/if}
```

Test Again

To test these changes you must test everything as above... but remember you are testing for numerous conditions.

1. Anonymous users should not be able to see the 'Restricted Test 1' page in the navigation.

2. Anonymous users should be redirected to the login page if attempting to navigate directly to the 'Restricted Test 1' page.
3. Logged in 'pending' users should not be able to see the 'Restricted Test 1' page
4. Logged in members of the 'pending' group should be redirected to the login page if attempting to navigate directly to the 'Restricted Test 1' page.
5. Logged in members of the 'members' group should see the 'Restricted Test 1' page
6. Logged in members of the 'members' user group should be able to navigate directly to the 'Restricted Test 1' page

Summary of Example 2

After successful examination of the page template, the menu manager template, and the testing illustrated above, you should now understand that

- Multiple 'mostly similar' page templates are mostly not necessary
- It is possible to hide content from certain users based upon user group membership.

Example 3: Hiding the Content

This example will remove the redirection that we have been using up to this date, and will allow the user to display all pages, including those marked as restricted, but will hide the restricted content from them.

Modifying the Page Template

If you recall, we have a simple expression in our 'Left Simple Navigation + 1 Column' page template that tests if the page being requested was marked as 'restricted', and if so, and the user was not a member of the 'members' user group, the user is redirected to the 'login' page.

In this example, we'll be removing removing that expression completely, and using similar logic to optionally display the {content} tag. This new expression will allow people to directly navigate to a page, even if it is marked as restricted, but will not show them any of our secure page content.

To do this:

- Find this logic in the “Left Simple Navigation + 1 Column” template, and remove it completely.

```
{page_attr key='extra1' assign='extra1'}

{if !$ccuser->memberof('members') and $extra1 == 'restricted'}
```

```
      {redirect_page page='login'}
    {/if}
```

- Find the {content} tag in the same template and replace it with:

```
    {page_attr key='extra1' assign='extra1'}
     {if ($ccuser->memberof('members') and $extra1 == 'restricted') or
         $extra1 == ''}
      {content}
    {else}
       <h4 style=”color: red;”>Attempt to access unauthorized content</h4>
    {/if}
```

Discussion about the {content} Block, CustomContent and inline forms

It has come time to talk about the forms and links generated by the various modules of CMS Made Simple (including forms like from the Search module, News module, and from the FrontEndUsers module itself). This is an important concept to learn, so please read carefully.

Essentially, when a developer is writing a module, and deciding upon the behavior for that modules links and forms, there is a decision to make for each link and each form that determines it's resulting behavior

1. Non-inline *(the default)*

   This means that the action to be called, will be called in place of what would normally would happen within the {content} tag.

2. Inline

   This means that the results of the form (if any) will replace the original tag.

Lets consider the template that we have been using throughout this document; “Left Simple Navigation + 1 Column”. We have just modified it such that when browsing to a restricted page, when not logged in, or a not a member of the 'members' user group, we will receive a nice red 'Attempt to access unauthorized content' message. And this is fine. But there are problems as well.

Case 1: Search

Try this:

Ensure that you are logged out of the FrontEndUsers module completely

Using the address bar on your navigator, enter the address:

http://<my cms install>/index.php?page=restricted-test-1

As we've discussed before, it will work and you will see the 'Attempt to access unauthorized content message'.

Enter a value into the Search field on the top of that page, and click 'Submit' on that form.

As you can see, the search module did not display any results. Why is this? Simply, it is because the search module is acting in 'non-inline' mode. And our current page template does not allow pages marked as 'restricted' *(remember the extra1 attribute stuff above, and the template changes you just made)* to display the {content} block you are a logged in member of an appropriate group.

If you were to navigate back to the “Home” page and try the same test, it would succeed. This is because the “Home” page fails the test that we just added to the page template.

Why is this important? Because considering this fact, care must be taken as to how a module behaves and therefore what, how, and where to place its call into your page template or page content, considering the security restrictions that we want on some content pages. You may want to reconsider your design based on the behavior (or maybe even limitations of some modules).

Case 2: News

The News module is used in the default page templates to generate a summary view in the sidebar. It also generates non-inline links to detail views of specific articles. But yet clicking on one of those links when currently logged out, and displaying the “Restricted Test 1” page works fine. Why? After close examination of the page template, we can see this: {news number='3' detailpage='news'}. After also reading the help to the news module we can understand that those links go to a different content page (one that is not marked as restricted). Whereby the Search module is not called in a similar manner to behave in the same way.

Case 3: FrontEndUsers

Often, the FrontEndUsers module is called from within the Page Template so that there is a convenient login prompt on all pages, and the user has convenient access to a 'logout' link, and to the ability to change settings. However, the logout, change settings, and lost username functionality all uses (by default) non-inlined code. This means that the resulting code will replace the {content} tag before it is executed... and our page template may deny that code from executing based upon the current logic.

As you can see, there are many considerations that must be made when creating pages with much functionality, but with restricted access. Each of these considerations can effect your template design, and also your layout. Some of those considerations include:

- What content is secure?
  - To whom?

    - Are there different levels of access?
- What modules should be called from the page template?
    - How should they behave for invalid users on restricted pages?
    - Is Non-Inline behavior okay?
    - Is Inline behavior okay?
    - Do the modules support the desired behavior?
- What modules will be called from page content on restricted pages?
    - How will they behave?

Summary of Example 3

We have learned how to allow users to browse to any page, even though they do not have permission to view the content on that page.

Additionally, we have learned that care must be taken when deciding on the behavior and appearance of the website because of the the fact that a restriction on the use of the {content} tag may have adverse behavior with modules that we want to use on the same page. Therefore we must carefully analyze the behavior of each module when considering the design of the site.

Example 4: Hiding Restricted Content Only

The final example is only a slight modification to the one above, in that we get past the problem of the search module or other 'necessary' modules that use non-inlined output, whilst still retaining the integrity of our secure content.

Modify The Page Template

In this example we will be introducing the use of "Additional Content Blocks". If you didn't know this already, CMS Made Simple has the ability to use provide additional areas where content or data associated with a page can be created. It can then be displayed in any way specified by your page template. Using just a slight modification to our existing page template we can use additional content blocks for our secure content, whilst maintaining the functionality of the Search and other modules which use non-inlined code.

Try This:

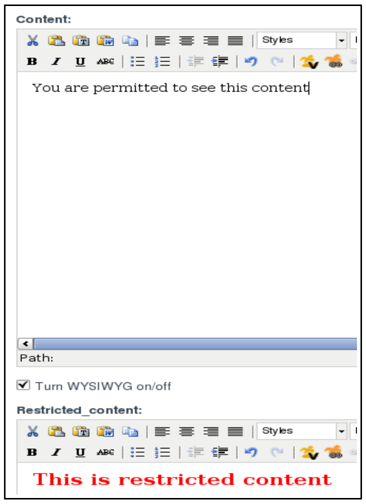
- Change the portion of the page template entitled "Left Simple Navigation + 1 Column" from:

```
{page_attr key='extra1' assign='extra1'}
{if ($ccuser->memberof('members') && $extra1 == 'restricted') || $extra1 == ''}
  {content}
{else}
  <h4 style="color: red;">Attempt to access unauthorized content</h4>
{/if}
```

to

```
{content}
{page_attr key='extra1' assign='extra1'}
{if ($ccuser->memberof('members') && $extra1 == 'restricted') || $extra1 == ''}
  <br/>{content block='restricted_content'}
{/if}
<br/>
```

- Edit the content of the “Restricted Test 1” page, you will now see two content blocks displayed.


Content:

You are permitted to see this content

Path:

☑ Turn WYSIWYG on/off

Restricted_content:

This is restricted content

- Copy the content from the main Content box into the second box
- Change the default content box to contain a smarty comment (see the image below).

Testing

To test the functionality of this page, you should repeat all of the tests for Example1, Example2, and Example3. Including the Search test when visiting the restricted page and not logged in.

You will notice that everything functions as before except that:

- The message “you are authorized to see this content” is displayed at all times.
- The search test works.

Summary of Example 4

Example 4 teaches us that there is a mechanism whereby advanced templates can be created that allow users to view a page, but not to view secured content within that page. And that this provides us with the capability to display different content to different users from within the same page.

Conclusion

After reading this document, you should have a basic understanding of

- The functions of the different modules:
    - FrontEndUsers

      Provides the login, logout functionality, management of frontend users, their properties, and user groups, and the ability to interface with other modules.
    - SelfRegistration

      Provides the ability for anonymous users to register themselves with your website.
    - CustomContent

      Provides the interface between the FrontEndUsers module and Smarty such that web developers can create websites that appear or behave differently for different users.
- Have a basic understanding of how to deploy the modules.
    - Install FrontEndUsers, and CustomContent
        - Create at least one user property
            - This must be done first before you can create a user group
            - There are numerous different types of properties, including Image, TextArea, and dropdown boxes
            - You do not have to create a property for username or password, these are automatically provided.
        - Create at least one user group
            - You must associate at least one property with a user group
            - You can associate numerous properties with each user group
            - Properties can be marked as Optional, Required, or Hidden when associated.
            - Properties can be re-ordered to adjust the order that they appear in when a user is registering, or adjusting their properties.
        - Manually create users in the FrontEndUsers module
            - Authorized Administrators can specify a users group membership, and adjust all user properties, including the password.
        - Setup FrontEndUsers preferences

        - Create a login page that handles login, logout, and forgot password functionality
    - Install SelfRegistration
        - Setup SelfRegistration preferences
        - Create a register page that allows users to register into a 'pending' group
- Restrict access to pages
    - You can signify pages as 'restricted' using the 'extra1' or any of the extra page properties.
    - You must disable caching on restricted pages
    - You can modify the page template to completely deny access to restricted pages
    - How to create a new navigation template to hide restricted the listing of restricted pages to anonymous users
    - How to create restricted access based on user group membership
    - How to modify the page template to hide restricted content on restricted pages.

Recommended Reading

It is recommended (almost required) that all users of CMS Made Simple take the time to read the following:

- The help that is included with each module that you are using, including the "core" modules.
- The help that is included with CMS Made Simple tags (under the "Extensions >> Tags" menu.
- Calguy's helpful blog at calguy1000.com
- The documentation in the CMS Made Simple Wiki (wiki.cmsmadesimple.org)
- The smarty documentation at Smarty.net

About the Author

Robert Campbell, the author of this document is also a senior development team member / project manager for CMS Made Simple, and the author of the CMSMS add-on modules described in this document.

Robert has over fifteen years of professional software development, customer support, and system administration experience

In 1992 after graduating from the Southern Alberta Institute of Technology he spent in excess of ten years working in the Oil and Gas industry working on CAEX (Computer Aided Exploration) software. This software was written primarily in C and C++ for both windows and unix. His duties ran the gambit from project management to software development, customer support, and large, distributed network administration.

In approximately 2004, whilst looking for a convenient way to create a website about his family information that was easy to maintain he stumbled across CMS Made Simple. It's friendly development team, open environment, and well built architecture was enough to hook him, and soon he was writing additional software to add on to CMSMS. Even though he had little experience in PHP, he was able to leverage his knowledge of C and C++, and was developing quickly. The rest is history.

Since that time, Robert has written and continues to enhance and maintain dozens of add-ons for CMS Made Simple, many of them are the most popular modules used by the community today. Robert has continued to contribute to the CMSMS Project, to the point where he now contributes many of the bug fixes and enhancements to the core, manages the testing and release cycles, and assists with the development and maintenance of the CMS Made Simple websites. Robert now works in CMS Made Simple full time, and takes on numerous contracting tasks related to CMS Made Simple.