Super Groups

Overview

Super groups are advanced HUB groups, that have their own webspace within the HUB to showcase their group.

Super groups have a lot of extra functionality built in to allow them to customize their group.

Group Pages & Modules

Super groups have the ability to include PHP and javascript code into group pages and modules. Pages or modules that contain PHP or Javascript code will then need to be approved by a group page approver. Notifications are sent to approvers when a page needs to be approved. Another notification will be sent to the group managers when the page has been approved.

Templating System

Overview

A new templating system has been added to help Super groups create a better web presence. When a super group is created, a default template is created and placed in the groups filespace.

The only file needed for a super group template to work is /{web_root}/site/groups/{group_id}/template/index.php

File Structure

Below shows the desired file directory structure for super groups. Following this pattern will allow HUB owners and developers to add new developments and find bugs easier.

Default Template

A default template is created for each super group. This can be used as a base for the super groups template.

Error Template

Super groups have the ability use a custom error template (error.php), which can include a stylesheet (error.css) or scripts to display a custom error page.

Template Includes

The following group include tags can be used within a template to display the content, the menu, the member/manager toolbar, modules, or include a Google Analytics tracking code.

- <group:include type="content" />
- <group:include type="content" scope="before" />
- <group:include type="menu" />
- <group:include type="toolbar" />
- <group:include type="modules" postion="{position}" />
- <group:include type="modules" title="{title}" />
- <group:include type="googleanayltics" account="{account}" />
- <group:include type="script" base="" source="{file_path}" />
- <group:include type="stylesheet" base="" source="{file_path}" />

For Script & Stylesheet group includes you can specify a base param of "template" which will automatically prepend "/template/assets/js" or "/template/assets/css" to the source. If no base is specified, it will look for the file in the groups "uploads" directory.

Page Templates

Overview

You'll probably want most of your group pages to look about the same. Sometimes, though, you may need a specific page, or a group of pages, to display or behave differently. This is easily accomplished with page templates.

Specialized Page Templates

Create a template for one Page: Intended for one specific page, you can create a specialized template, named with that page's alias or ID:

1. page-{alias}.php
2. page-{id}.php

For example: Your About Us page has an alias of 'about-us' and an ID of 6. If template has a file named page-about-us.php or page-6.php, then it will automatically find and use that file to render the About Us page.

To be used, specialized page templates must be in your groups template directory: /{web_root}/site/groups/{group_id}/template/

Custom Page Templates

Create a template that can be used by any page: A custom page template can be used by multiple pages. To create a custom page template make a new file starting with a template name inside a PHP comment. Here's the syntax:

```
<?php
/*
Template Name: My Custom Page
*/
```

To be used, custom page templates must be in your groups template directory: /{web_root}/site/groups/{group_id}/template/

Selecting a Page Template

Once you upload the file to your template's folder, the template name, "My Custom Page", will list in the edit page screen's Template dropdown.

Template Hierarchy

The order below defines which page template gets loaded on any given page. The first match found is used.

1. Custom Template â€” If the page has a custom template assigned, the HUB will looks for that file and, if found, use it.
2. page-{alias}.php â€” Else the HUB looks for and, if found, uses a specialized template named with the page's alias.
3. page-{id}.php â€” Else the HUB looks for and, if found, uses a specialized template named with the page's ID.
4. page.php â€” Else the HUB looks for and, if found, uses the default page template.
5. index.php â€” Else the HUB uses a the template's index file.

Page Includes

The following group include tags can be used within a group page.

- <group:include type="modules" postion="{position}" />
- <group:include type="modules" title="{title}" />
- <group:include type="script" base="" source="{file_path}" />
- <group:include type="stylesheet" base="" source="{file_path}" />

For Script & Stylesheet group includes you can specify a base param of "template" which will automatically prepend "/template/assets/js" or "/template/assets/css" to the source. If no base is specified, it will look for the file in the groups "uploads" directory.

Custom Macros

Overview

Super groups have the ability to create their own custom macros or override any existing macro [[MacroName(args)]].

Custom Macro Class Structure

```
<?php
/**
 * HUBzero CMS
 *
 * Copyright 2005-2014 Purdue University. All rights reserved.
 *
 * This file is part of: The HUBzero(R) Platform for Scientific Collab
oration
 *
 * The HUBzero(R) Platform for Scientific Collaboration (HUBzero) is f
ree
 * software: you can redistribute it and/or modify it under the terms
of
 * the GNU Lesser General Public License as published by the Free Soft
ware
 * Foundation, either version 3 of the License, or (at your option) an
y
 * later version.
 *
 * HUBzero is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public Li
cense
 * along with this program.  If not, see .
 *
 * HUBzero is a registered trademark of Purdue University.
 *
 * @package   hubzero-cms
 * @copyright Copyright 2005-2014 Purdue University. All rights reserv
ed.
 * @license   http://www.gnu.org/licenses/lgpl-3.0.html LGPLv3
 */
```

```
namespace PluginsContentFormathtmlMacros;

use PluginsContentFormathtmlMacro;

/**
 * Wiki macro class for displaying hello world
 */
class {{macro_name}} extends Macro
{
 /**
  * Returns description of macro, use, and accepted arguments
  *
  * @return     array
  */
 public function description()
 {
  $txt = array();
  $txt['html'] = '

Put macro description here...

';
  return $txt['html'];
 }

 /**
  * Generate macro output
  *
  * @return     string
  */
 public function render()
 {
  return 'Return any html content you want this macro to render';
 }
}
```

Overriding Macros

To override a macro, copy the original macro located in /{web_root}/plugins/content/formathtml/macros/ into your groups macro folder /{web_root}/site/groups/{group_id}/macros/. You can no modify the render functionality, add or

remove params, etc.

Note: The Macos class name must remain the same and the class must implement a render() method. You are free to add or chanage other methods.

Note: If the original macro file is located within a subfolder, you must recreate that folder structure in the groups macros folder for the override to work.

PHP Pages

Overview

Super groups have the ability to include PHP code in any group page or module through the page and module managers. If you are finding this is hard to manage or the approval process is taking too long. Users with SSH access and PHP knowledge can add any number of PHP pages to their super group.

PHP Pages Directory

/{web_root}/site/groups/{group_id}/pages/

PHP Page Hierarchy

- /{web_root}/site/groups/{group_id}/pages/features.php -> /groups/{group_cn}/features
- /{web_root}/site/groups/{group_id}/pages/features/one.php -> /groups/{group_cn}/features/one
- /{web_root}/site/groups/{group_id}/pages/features/two.php -> /groups/{group_cn}/features/two

PHP Page Includes

The following group include tags can be used within a PHP page.

- <group:include type="modules" postion="{position}" />
- <group:include type="modules" title="{title}" />
- <group:include type="script" base="" source="{file_path}" />
- <group:include type="stylesheet" base="" source="{file_path}" />

For Script & Stylesheet group includes you can specify a base param of "template" which will automatically prepend "/template/assets/js" or "/template/assets/css" to the source. If no base is specified, it will look for the file in the groups "uploads" directory.

Databases

Overview

Each super group comes with its own database. This database can be used to store data for that group. The credentials for accessing that database can be found in the super groups database config file.

Config Path

/{web_root}/site/groups/{group_id}/config/db.php

Config File Contents

```
<?php
return array(
 'host' => 'localhost',
 'port' => '',
 'user' => 'sgmanager',
 'password' => 'xxxxx',
 'database' => 'sg_{group_cn}',
 'prefix'   => ''
);
```

Using the Database

You can use the database anywhere you want in your template, a PHP page, a group component, etc. Anywhere you can run PHP code basically.

Getting a reference to the group database object is very easy:

```
$database = HubzeroUserGroupHelper::getDBO();
```

You can access the group database and the HUB database at the same time. Use the above call to get access to the group database and JFactory::getDBO(); to get access to the HUB database. All you have to do is store them in two different variables.

Migrations

Overview

Migrations allow you a group update its separate database without having to connect to the live database and manually updating the schema. Another benefit to using migrations is that they are automatically run every time the super groups code is updated from Gitlab!

Creating a Migration

Migrations can be created easily with the HUbzero command line application "Muse". From the command line run the following command (in the web root):

/{web_root}/cli/muse.php group scaffolding migration --group={group_cn} -e=com_{component}

Simply replate {group_cn} with your groups cname and {component} with the component. The migration file will be automatically placed into the correct location, ready for you to modify and commit when ready.

Running Migrations

Running migrations is almost as easy as creating them with once again help from the Hubzero command line application. From the command line run the following command (in the web root):

/{web_roo}/cli/muse.php group migrate -if --group={group_cn}

Simply replate {group_cn} with your groups cname. The -i argument means ignore dates (run migrations it could have missed) and -f means actually run.

Note: You only need to manually run migrations in a dev environment. When groups code is updated on live, migrations are automatically run.

Components

Overview

Super groups have the ability to have their own components. The structure of a super group component is largely the same as a full CMS component with a few small differences. The primary difference related to the different application environments or types a full CMS component can support. While a full component may have controllers, views, and logic for site, admin, api, a super group component **only** has code for the equivalent of site.

Here's an example of CMS components vs super group components structure:

```
// CMS Component
/components
.. /com_example
.. .. /admin
.. .. .. /controllers
.. .. .. .. records.php
.. .. .. /language
.. .. .. .. /en-GB
.. .. .. .. .. en-GB.com_example.ini
.. .. .. /views
.. .. .. .. /records
.. .. .. .. .. /tmpl
.. .. .. .. .. .. display.php
.. .. .. example.php
.. .. /config
.. .. .. config.xml
.. .. /helpers
.. .. .. html.php
.. .. /models
.. .. .. record.php
.. .. /site
.. .. .. /controllers
.. .. .. .. one.php
.. .. .. /language
.. .. .. .. /en-GB
.. .. .. .. .. en-GB.com_example.ini
.. .. .. /views
.. .. .. .. /one
.. .. .. .. .. /tmpl
.. .. .. .. .. .. display.php'
.. .. .. example.php
.. .. .. router.php
.. .. example.xml
```

```
// Super Group Component
/components
.. /com_example
.. .. /controllers
.. .. .. one.php
.. .. /helpers
.. .. .. html.php
.. .. /models
.. .. .. record.php
.. .. /language
.. .. .. /en-GB
.. .. .. .. en-GB.com_example.ini
.. .. /views
.. .. .. /one
.. .. .. .. /tmpl
.. .. .. .. .. display.php
.. .. example.php
.. .. example.xml
.. .. router.php
```

Here you can see that the contents of the super group component is the same as the /site sub-directory of the CMS component with the addition of the XML manifest and /helpers & /models directories. With include path updated, this covers the bulk of the changes. Views, models, routing, controllers, and so on should handled int he same manner as full CMS components.

For more information regarding developing components see:
https://hubzero.org/documentation/1.3.0/webdevs/components

Components Directory

/{web_root}/site/groups/{group_id}/components/com_{component}/

Component Language Files

/{web_root}/site/groups/{group_id}/language/en-GB/en-GB.com_{component}.ini

Component Paths

As a helper for super group component developers the path to the component directory is

defined in a constant.

JPATH_GROUPCOMPONENT

So as an example, if your creating the component "com_drwho", the JPATH_GROUPCOMPONENT constant equals:

/{web_root}/site/groups/{group_id}/components/com_drwho/

Note: You should be able to move the component to the main components folder and and have it work without any changes.

URL's built within a super group component will automatically have "/groups/{group_cn}/" prepended to them. Please don't manually do that in your component or it will result in an error.

Creating a Component

Creating components can always be done manually by creating the files in the correct location as described above. You can also utilize the Hubzero command line application. From the command line run the following command (in the web root):

/{web_root}/cli/muse.php group scaffolding component --group={group_cn} -n=com_{component}

Simply replate {group_cn} with your groups cname and {component} with the component. The component files will be automatically placed into the correct location, ready for you to modify and commit when ready.