Commands

Cache

The cache command is a helper for clearing your sites cache files. You can clear the entire cache, or just the CSS cache. Those commands, respectively, are:

```
php muse cache clear

php muse cache:css clear
```

Configuration

The configuration command is used to personalize and customize your Muse experience. It's also used to store variables for repeated use. For example, the scaffolding command will ask you, if you haven't already, to set your name and email to be used when generating files.

```
muse configuration set --user_name="John Doe"
muse configuration set --user_email=john.doe@gmail.com
```

Configuration can also be used to store hooks and aliases. Hooks are additional commands that are run at pre-defined points. Aliases are command shortcuts. Here are some examples:

```
# run permissions fix after updating the repository
muse configuration:hooks add repository.afterUpdate "chmod -R g+w /www
/docroot"

# Add a shortcut for the environment command
muse configuration:aliases add env environment
```

Database

The database command was added for two primary reasons - the first backups, and the second, reverse content migration. Backups are fairly straight-forward, but a little more detail is in order for reverse content migration.

If you have an environment with more than one stop in your production flow, you've likely run into the problem of wanting to move data from prod to dev for testing purposes. But in so doing, you often overwrite some site-specific configuation on dev. So get around this, we perform a dump and load using the database command to move only those things that should move between environments.

```
# dump the database
muse database dump

# then make sure you copy to your dev environment
# then from dev, load the dump back up (it will have a different name)
muse database load filenamefromabovecommand
```

Environment

The environment command simply outputs the current environment variables.

```
Current user     : Mr Awesome <awesome@gmail.com>
Current database : example
```

Extension

If you don't already know, extensions are the general name for all of the 'apps' allowed by the HUBzero framework. They include (amoung some others), templates, components, modules, and plugins. When adding a new extension, you will often want to add it to the extensions database table and enable it. This command can help save you trips directly to the database.

The nice thing to about the extension command is that it will prompt you for what it needs, you don't really need to remember the syntax.

```
me@me.org:~# muse extension
What do you want to do? [add|delete|install|enable|disable] add
What extension were you wanting to add? com_awesome
Successfully added com_awesome!
```

Or, as another example. Let's delete that entry we added above using the written out syntax

```
me@me.org:~# muse extension delete --name=com_awesome
Successfully deleted com_awesome!
```

Note that if you're in a production environment and using migrations, this command is redundent. Use migrations! But if you're just testing and need a quick way to enable or disable something, this is the way to go.

Group

The group commands are simply wrappers on existing commands to be used within the super group context. Please review the super group documentation for more details.

Log

The log command is great for following and filtering log entries. There are currently two log types available, the profile log and the query log. To start, simply:

```
muse log follow profile
```

You have to having logging enabled for new entries to be displayed!

Once started, you'll see info on the log fields being displayed.

```
me@me.org:~# muse log follow profile
The profile log has the following format (* indicates visible field):
<0:*timestamp> <1:*hubname> <2:*ip> <3:*app> <4:*uri> <5:*quer
y> <6:*memory> <7:*querycount> <8:*timeinqueries> <9:*totaltime>
```

To toggle a fields visibility, simply press the number next to the field of interest. For example, pressing 2, and then f to show the fields again, results in:

```
> Hiding ip
> The profile log has the
following format (* indicates visible field):
<0:*timestamp> <1:*hubname> <2:ip> <3:*app> <4:*uri> <5:*quer
y> <6:*memory> <7:*querycount> <8:*timeinqueries> <9:*totaltime>
```

To show the available commands, simply type h.

```
> q: quit, h: help, i: input mode, p: pause/play, b: beep on/off, f: f
ields, r: rerender last 100 lines
```

Migration

For more info on the migration command, see the dedicated migrations section under the database chapter.

Repository

The repository command offers an abstraction on top of the mechanism used to manage and update the CMS. This could include GIT, HTTP-based package installs, or Debian packages. Currently, GIT is the only supported mechanism, but more are to come in the future.

To start, simply see if the repository command is supported in your environment.

```
me@me.org:~# muse repository
This repository is managed by GIT and is clean
```

If you environment is not currently supported, you'll receive a message like this:

```
me@me.org:~# muse repository
Sorry, this command currently only supports setups managed by GIT
```

To start the update process, use the update task. Depending on your current state, you'll either see that you're up-to-date, or see what's coming in the next update.

```
me@me.org:~# muse repository update
The repository is already up-to-date
```

or...

```
me@me.org:~# muse repository update
The repository is behind by 747 update(s):
...
```

Then, to perform the actual update, add the -f flag.

```
me@me.org:~# muse repository update -f
Updating the repository...complete
```

If something goes wrong, the update mechanism will automatically roll back to it's state prior to attempting the update. Then you'll have to go in a manually perform the update depending on the mechanism.

Spring Cleaning

In addition to performing updates, the repository command also offers some help doing periodic cleanup. Using the clean command will allow you to prune rollback points and stashes.

```
me@me.org:~# muse repository clean
Do you want to purge all rollback points except the latest? [y|n] y
Purging rollback points.
Do you want to purge all stashed changes? [y|n] y
Purging repository stash.
Clean up complete. Performed (2/2) cleanup operations available.
```

Scaffolding

Scaffolding was create to help developers get started quickly. Let's be honest, developers rarely start from a blank file. We copy something existing and modify. With scaffolding, we give you a template a pre-fill known values to make this process even easier.

At this time, scaffolding knows how to create:

- Commands
- Components
- Migrations
- Tests

So, for example, to create a new component, simply:

```
me@me.org:~# muse scaffolding create component com_awesome
Creating /var/www/example/core/components/com_awesome/awesome.xml
Creating /var/www/example/core/components/com_awesome/admin/awesome.ph
p
Creating /var/www/example/core/components/com_awesome/admin/controller
s/awesome.php
Creating /var/www/example/core/components/com_awesome/admin/language/e
n-GB/en-GB.com_awesome.ini
Creating /var/www/example/core/components/com_awesome/admin/language/e
n-GB/en-GB.com_awesome.sys.ini
Creating /var/www/example/core/components/com_awesome/admin/views/awes
ome/tmpl/display.php
Creating /var/www/example/core/components/com_awesome/api/controllers/
api.php
Creating /var/www/example/core/components/com_awesome/config/access.xm
l
Creating /var/www/example/core/components/com_awesome/config/config.xm
l
Creating /var/www/example/core/components/com_awesome/models/awesomes.
php
Creating /var/www/example/core/components/com_awesome/site/awesome.php
Creating /var/www/example/core/components/com_awesome/site/assets/css/
awesome.css
Creating /var/www/example/core/components/com_awesome/site/assets/js/a
wesome.js
Creating /var/www/example/core/components/com_awesome/site/controllers
/awesome.php
Creating /var/www/example/core/components/com_awesome/site/language/en-
GB/en-GB.com_awesome.ini
Creating /var/www/example/core/components/com_awesome/site/router.php
Creating /var/www/example/core/components/com_awesome/site/views/aweso
mes/tmpl/display.php
Creating /var/www/example/core/components/com_awesome/site/views/aweso
```

```
mes/tmpl/edit.php
```

As you can see, this automatically generates all of the core files and views you're likely to need. It also names them appropriately, as well as using the provided component name to even tweak the contents of these files.

Test

Testing is critical to both deploying a new extension, and updating existing extensions without too much heartache. To facilitate testing, muse offers a framework and wrapper around the popular PHP Unit testing infrustucture.

To see the current extensions with tests, run:

```
me@me.org:~# muse test show
lib_database
```

Then, to run a specific extensions tests, you can use the run command.

```
me@me.org:~# muse test run lib_database
PHPUnit 4.6.2 by Sebastian Bergmann and contributors.

..................................................

Time: 2.26 seconds, Memory: 17.5Mb

OK (51 tests, 73 assertions)
```

User

The final command available at this time is the user command. It offers some advances administrative functionality for merging and unmerging users.

This command is experimental!

Occasionally, on a hub, one person will create two accounts and not realize it. They later ask

you to merge the accounts and move the contributions from one to the other. This isn't a simple task, and involves updating many, many references in the database. Fortunately for you, we've been working on a solution.

```
me@me.org:~# muse user merge 1042 into 1003
Updating (1) item(s) in jos_collections.object_id
Updating (1) item(s) in jos_collections.created_by
Updating (1) item(s) in jos_collections_items.created_by
Updating (8) item(s) in jos_courses_asset_groups.created_by
Updating (15) item(s) in jos_courses_assets.created_by
Updating (1) item(s) in jos_courses_members.user_id
Updating (2) item(s) in jos_courses_offering_section_dates.created_by
Updating (2) item(s) in jos_courses_units.created_by
Updating (76) item(s) in jos_developer_access_tokens.uidNumber
Updating (1) item(s) in jos_developer_applications.created_by
Updating (1) item(s) in jos_developer_rate_limit.uidNumber
Updating (9) item(s) in jos_users_log_auth.user_id
Ignoring jos_users_password.user_id due to integrity constraint violat
ion
Updating (1) item(s) in jos_users_points.uid
Ignoring jos_xprofiles_bio.uidNumber due to integrity constraint viola
tion
Updating (3) item(s) in jos_xprofiles_tokens.user_id
```

Then, if needed, you can reverse the merge.

```
me@me.org:~# muse user unmerge 1042 from 1003
Unmerged (122/122) records successfully!
```