

# Migrations

## Overview

HUBzero offers the muse command for automating and simplifying common web developer and system administrator tasks. Of those tasks, running database and content migrations is probably the most crucial to the successful management and deployment of new and updated HUBzero extensions.

The following sections assume that you have the muse command in your path and can execute the script directly. If that is not the case, replace all calls to muse with `/www/yourdocroot/cli/muse.php`.

In addition to this documentation, more detailed documentation can always be found by calling: `muse migration help`

## Running Migrations

Running migrations in its basic form is rather simple (though there are a plethora of options available to complicate things if you so desire). Simply type `muse migration` to run migrations in dry-run mode. This will tell you if you have any pending migrations to run, or if you have perhaps missed a previous migration. If satisfied with what migrations thinks needs to be done, simply run `muse migration -f` to run the full migration.

That's it!

OK, there's more. By default, migrations won't run migrations that have been missed in the past. To tell migrations to run all pending migrations, irrelevant of date, include the `-i` option. All other available options can be found by running `muse migration help` as mentioned above.

## Creating Your Own Migrations

This is where the fun begins...

Creating migrations is essential to anyone deploying new extensions in an environment where database tinkering on prod is frowned upon. If the idea of production database access doesn't send chills down your spine, then at least having a migration written will offer a well documented change log for your extensions.

Muse has some basic commands for scaffolding, one of which allows you to create a template migration. To get this auto-generated goodness for yourself, type `muse scaffolding create migration -e=extension_name`. Here, the extension name would be the extension you are working on, in the form of `com_mycomponent` or `plg_stuff_coolthing`. This will drop you into your

default editor with the template migration in place and setup according to the HUBzero conventions of naming and layout.

The migration command will manage what migrations have been run and in what environment. That way you don't have to worry about what you've run and where. That being said, we think it's generally a good idea to make your migrations as foolproof and backwards compatible as possible. To that end, we've added a handful of helper functions to make things as simple as possible. These functions are available on the database object inside of your migration. They are as follows:

- `$this->db->ifTableExists('tableName');`
- `$this->db->ifTableHasField('tableName', 'fieldName');`
- `$this->db->ifTableHasKey('tableName', 'keyName');`

As an example, instead of just blindly executing an alter table statement to add a new column, you might instead wrap the execution of that statement in an if block that checks for the existence of the table, and the non-existence of the field you want to add...like so:

```
if ($this->db->tableExists('myTable') && !$this->db->tableHasField('myTable', 'myNewField')) :
```

Feel free to glance at other migrations in `/www/your_doc_root/migrations` for sample usage.

We've also started adding some additional features to make generating your migrations even easier. So, for example, if you're writing a migration to generate a new table, you can now do `muse scaffolding create migration for jos_table_name -e=extension_name`. This will create the migration as before, but this time, the migration is completely written for you! We'll add more info here as new features are developed. Also note, the extension name is still required at this time, as the table name and extension name are not explicitly related.