

Building Your First Module



Dennis Jarecke

Drupal Camp Ohio

November 15, 2014

Drupal

The Goal

Build a module from scratch that will do the following:

- Create a page
- Create a form
- Create a database table
- Select/insert form data from/into database

And then I will ramble on about things like:

- Logging
- Development Module
- Implementing Javascript
- Accessing Node Data
- Drupal Search



Get My Code and Presentation!

<https://github.com/tetmo113/buildyourfirstmodule>



Introducing api.drupal.org

API reference

[Drupal 4.6](#) [Drupal 4.7](#) [Drupal 5](#) [Drupal 6](#) **[Drupal 7](#)** [Drupal 8](#)

Welcome to the Drupal developer's documentation. Newcomers to Drupal development should read the conceptual information provided in the "A few components of Drupal" section below, and then proceed to examine one of the heavily-documented example modules below. The examples are fully-functioning Drupal modules, so you can download them from the contributions repository and alter them as you experiment.

- A few components of Drupal
 - [Module system \(Drupal hooks\)](#)
 - [Database abstraction layer](#)
 - [Menu system](#)
 - [Form generation](#)
 - [File upload system](#)
 - [Field API](#)
 - [Search system](#)
 - [Node access system](#)
 - [Theme system](#)
 - [Constants](#)
 - [Global variables](#)
- Example modules
- In-depth discussions
 - [Forms API Reference](#)

You should browse these sections to get a good overview of what the Drupal API components do.



The Hook System

- A hook is a point in execution where Drupal seeks input from its modules.
- The name of the hook determine where it gets executed.
- Think of hooks as a set of predetermined functionality you can use.
- A module can use a hook function by using its machine readable name+_+ the hook name.
- A hook will be your main interaction with the Drupal system.
- Several examples will follow . . .



Where to put your module?

- Never put it in /modules! This is for core and your module will be removed when you upgrade core.
- /sites/all/modules - will work and is the place to put it for all sites if you have a multisite setup.
- /sites/all/modules/custom - preferred because you name space where your custom modules go.
- /sites/all/modules/contrib should be where contributed modules go.
- /sites/my_site_name/modules for site specific modules



Exercise 1: The Simplest Module

- Put `first_mod.info` in `/sites/all/modules/foo`

```
name = Build Your First Module Demo  
core = 7.x
```

- Put empty `first_mod.module` in `/sites/all/modules/foo`
- Note there are two module names:
 - Human readable defined by “name =” in `first_mod.info`
 - Machine readable and defined by name of `.info` file
 - Directory name plays no role in the module outside of the path to the module



Exercise 2: Modules Admin Screen

Let's configure how it looks on the modules screen by adding the following to `first_mod.info`:

```
package = Drupal Camp Ohio  
description = A really cool presentation on building your first module
```

Which produces on the module admin screen:



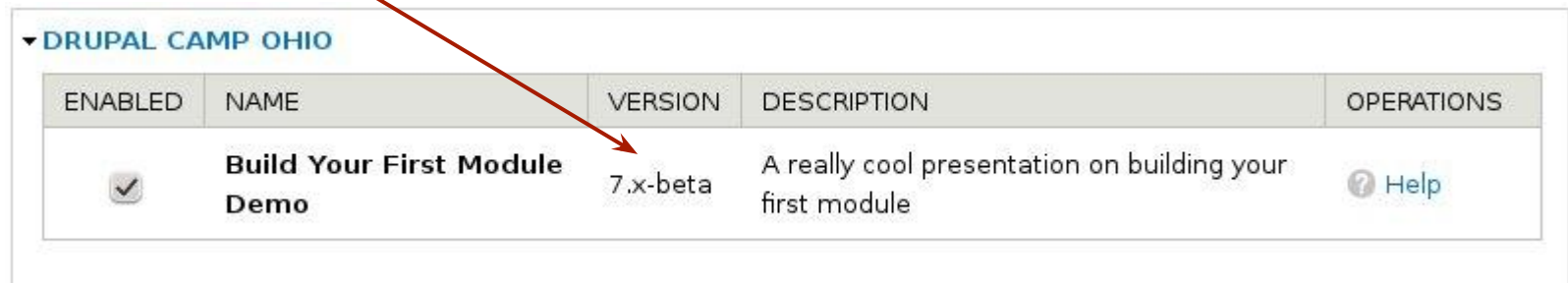
The screenshot shows the Drupal Modules Admin screen. A red arrow points from the `package = Drupal Camp Ohio` line in the code block above to the 'DRUPAL CAMP OHIO' header in the table. Another red arrow points from the `description = A really cool presentation on building your first module` line to the 'DESCRIPTION' column of the table row.

ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
<input checked="" type="checkbox"/>	Build Your First Module Demo		A really cool presentation on building your first module	



Exercise 2: cont'd

Now add a version with the line “version=7.x-beta”



A screenshot of the Drupal administration interface showing a table of modules. The table has columns for 'ENABLED', 'NAME', 'VERSION', 'DESCRIPTION', and 'OPERATIONS'. A red arrow points to the 'VERSION' column of the first row, which contains '7.x-beta'.

ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
<input checked="" type="checkbox"/>	Build Your First Module Demo	7.x-beta	A really cool presentation on building your first module	? Help



Exercise 2: cont'd

Add help with **hook_help**: add the function `first_mod_help()` to `first_mod.module`

```
<?php
// hook_help
function first_mod_help($path,$arg){
  switch($path){
    case 'admin/help#first_mod':
      $output = "This is the help system";
      $output .= "this is more help";
      return $output;
  }
}
```

Add `<?php` to beginning of `first_mod.module` but not `?>` to the end. This is the Drupal way!

See https://api.drupal.org/api/drupal/modules!system!system.api.php/function/hook_help/7 for adding help to another path in the module.

There are many `hook_helps`, but we use the functionality not by calling “`hook_help`”, but by replacing “`hook`” with our machine readable name and creating the function `first_mod_help`.

▼ DRUPAL CAMP OHIO

ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
<input checked="" type="checkbox"/>	Build Your First Module Demo	7.x-beta	A really cool presentation on building your first module	? Help



Exercise 3: A Simple Page

- **hook_menu**

- The name is deceiving. It's not just about a menu item
- **It's main role is to map a path with a function**

Let's demonstrate with a simple example . . .



Exercise 3: cont'd

```
function first_mod_menu(){
  $items['foo1'] =
    array(
      'title' => 'A simple page',
      'page callback' => 'foo1_page',
      'access callback' => TRUE,
      'type' => MENU_CALLBACK,
    );

  return $items;
}

function foo1_page(){
  $markup = '';
  $markup .= "<h1>Heading 1</h1>";
  $markup .= "<p>And now a paragraph</p>";
  return array('#markup' => $markup);
}
```

Found at /?q=foo1 or /foo1 with clean URLs

Required: title of page

Calls PHP function foo1_page()

Anyone can see this! Will demo hook_permission to change who can access this.

This is a simple page, but you can use a type of MENU_NORMAL_ITEM to create a menu item in Navigation that can be moved and hidden by admin. Use 'menu_name' to put it into a menu other than Navigation.

A lot could probably be said here on what you return, but this will get you going quickly.



Exercise 3: cont'd

```
$items['foo2'] =  
  array(  
    'title' => 'A simple page with permissions',  
    'page callback' => 'foo2_page',  
    'access callback' => 'user_access',  
    'access arguments' => array('access demo'),  
    'type' => MENU_CALLBACK,  
  );
```

Replaced TRUE with 'user_access'. Drupal function to determine a user's permission. Just do it!

hook_permission

```
function first_mod_permission(){  
  return  
    array(  
      'access demo' =>  
        array(  
          'title' => t('Access Drupal Camp Demo')  
        )  
    );  
}
```

From "name=" in .info file.

Build Your First Module Demo

Access Drupal Camp Demo



Exercise 4: Page Arguments

```
$items['color'] =  
  array(  
    'title' => 'Page argument example',  
    'page callback' => 'page_color_arg',  
    'page arguments' => array(1),  
    'access callback' => 'user_access',  
    'access arguments' => array('access demo'),  
    'type' => MENU_CALLBACK,  
  );
```

Will need another variable in
page_color_arg if have array
(2)

Everything between / not in the path can
be an argument. First variable in function
matches first thing between / and / not in
path. For example, try /color, /color/red
and /color/red/blue.

```
function page_color_arg($color){  
  $markup = '';  
  if( empty($color) ){  
    $markup .= "No color!";  
  }else{  
    $markup .= "The color is $color";  
  }  
  return array('#markup' => $markup);  
}
```



Exercise 5: Form Creation and Submission

Let's create a form page that will collect the following information:

- First name
- Email address

Our submit button will take us to another page where it will simply echo the input. (In Exercise 5 we will use this exercise to input the data into a database table.)



Exercise 5: An Overview of Forms

- Forms are weird. (My humble opinion.)
- But they have built in security protection. In particular against XSS.
- A form is a big array returned by a function - weired, but will demo it shortly.
- Our page defined by hook_menu will call `drupal_get_form` then render it with `drupal_render` and return it as markup as usual.

Let's demo it now . . .



Exercise 5: Create the form page

```
$items['myform'] =  
  array(  
    'title' => 'A simple form demo',  
    'page callback' => 'myform_page',  
    'access arguments' => TRUE,  
    'type' => MENU_CALLBACK,  
  );
```

Entry in hook_menu

drupal_get_form calls the function myformelements in this module. See the next page.

Render the form to HTML and return it along with other markup you want on the page.

```
function myform_page(){  
  $markup = 'Here is our form:<br>';  
  $foo = drupal_get_form('myformelements');  
  $markup .= drupal_render($foo);  
  return array('#markup' => $markup);  
}
```



Exercise 5: Creating the form

```
function myformelements(){
  $form = array();

  $form['first_name'] = array(
    '#type' => 'textfield',
    '#field_prefix' => 'Enter your first name:',
    '#description' => 'Put Bob if your name is Bob',
    '#weight' => 1,
    '#attributes' => array(
      'placeholder' => t('Bob')
    ),
  );

  $form['email'] = array(
    '#type' => 'textfield',
    '#field_prefix' => 'Enter your email:',
    '#weight' => 2,
    '#attributes' => array(
      'placeholder' => t('bob@example.com')
    ),
  );

  $form['submit'] = array(
    '#type' => 'submit',
    '#value' => 'Put whatever you want here',
    '#weight' => 3,
  );
}
```

The argument to `drupal_get_form` on the previous page.

A simple form demo

Here is our form:

Enter your first name:

Put Bob if your name is Bob

Enter your email:

How Drupal renders it:

```
<span class="field-prefix">Enter your first name:</span> <input placeholder="Bob" type="text" id="edit-first-name" name="first_name" value="" size="60" maxlength="128" class="form-text" />
<div class="description">Put Bob if your name is Bob</div>
```



Exercise 5: Validate and Submit

This is the other part of function `myformelements()`:

```
$form['#validate'][] = 'my_validate_function';  
$form['#submit'][] = 'my_submit_function';  
  
return $form;
```

- We must provide a function to validate and submit the data.
- See the next page for details . . .



Exercise 5: Validate Function

```
function my_validate_function($form,&$form_state){
  $email = $form_state['values']['email'];
  if( preg_match("/@/", $email) == 0 ) {
    form_set_error('email',t('Oops! You need an @ symbol!'));
  }
}
```

form_set_error will highlight and show error message that you want!



- Warning: Invalid argument supplied for foreach() in menu_unserialize() (line 400 of /home/emaiddemo/public_html/includes/menu.inc).
- Oops! You need an @ symbol!
- Warning: Invalid argument supplied for foreach() in menu_unserialize() (line 400 of /home/emaiddemo/public_html/includes/menu.inc).

This is extremely flexible! You can do any validation you want here before anything gets submitted.

Home



Navigation

▶ Add content

A simple form demo

Here is our form:

Enter your first name:

Put Bob if your name is Bob

Enter your email:

Put whatever you want here



Exercise 5: Submit Function

```
function my_submit_function($form,&$form_state){  
  $_SESSION['drupalcamp']['email'] = $form_state['values']['email'];  
  $_SESSION['drupalcamp']['first_name'] = $form_state['values']['first_name'];  
  $form_state['redirect'] = 'myform_result';  
}
```

`$_SESSION` variable stores the form data for retrieval by submit function. Probably several ways to do this, but this is how I do it.

```
$items['myform_result'] =  
  array(  
    'title' => 'Form submission results',  
    'page callback' => 'myform_results_page',  
    'access callback' => 'user_access',  
    'access arguments' => array('access demo'),  
    'type' => MENU_CALLBACK,  
  );
```

```
function myform_results_page(){  
  $email = $_SESSION['drupalcamp']['email'];  
  $first_name = $_SESSION['drupalcamp']['first_name'];  
  $markup = "First name: $first_name";  
  $markup .= "<br>Email: $email";  
  return array('#markup' => $markup);  
}
```

See <http://php.net/manual/en/reserved.variables.session.php> for info on `$_SESSION` variable.



Exercise 6: Database Tables

- We will show how to create a database table with Drupal's API
- We will use our form data in Exercise 4 to populate the table



Exercise 6: hook_schema

- Create file first_mod.install
- hook_schema will create the database tables
- Put it in first_mod.install
- See <https://api.drupal.org/api/drupal/includes!database!schema.inc/group/schemaapi/7> for lots of info
- Let's look at our hook_schema . . .



Exercise 5: Our first_mod_schema

```
function first_mod_schema(){
  $schema['myform_data'] =
    array(
      'description' => 'Stores form input data like email and first name.',
      'fields' => array(
        'id' => array(
          'type' => 'int',
          'unsigned' => TRUE,
          'not null' => TRUE,
          'default' => 0,
          'description' => 'The ID',
        ),
        'email' => array(
          'type' => 'text',
          'not null' => TRUE,
          'size' => 'big',
          'description' => 'Their email',
        ),
        'first_name' => array(
          'type' => 'text',
          'not null' => TRUE,
          'size' => 'big',
          'description' => 'Their first name',
        ),
      ),
      'primary key' => array('id'),
    );
  return $schema;
}
```

Table name

Columns

When installed it creates the table.

```
mysql> describe myform_data;
```

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	0	
email	longtext	NO		NULL	
first_name	longtext	NO		NULL	

```
3 rows in set (0.00 sec)
```



Warning!

If you are developing the table and have the module installed then you need to disable AND uninstall first! Only then can you enable it and have the table created.

**DON'T FORGET TO UNINSTALL YOUR
MODULE WHEN TESTING HOOK_SCHEMA!**



Exercise 6: DB Select

```
$query = db_select('myform_data', 'md');  
$query->fields('md', array('id'));  
$ids = $query  
->execute()  
->fetchCol();  
  
$max_id = max($ids);  
$markup .= "<br>max id:$max_id";
```

select id from myform_data as md

A trivial example but joins, where, order by, group by, aggregate functions, limit, distinct, etc. can all be done with Drupal db_select. Also you can write pure SQL and input that with db_query_range.

- fetchCol() returns a single column as indexed array
- fetchField() returns a single field from the next record
- fetchAssoc() returns next row as an associative array
- fetchAllAssoc() returns entire result set as associative array keyed by a given field
- fetchAllKeyed() returns entire result set as a single associative array

See <https://api.drupal.org/api/drupal/includes!database!database.inc/interface/DatabaseStatementInterface/7> on the types of outputs of db_select()->execute()



Exercise 6: DB Insert

```
$id = $max_id+1;

$query = db_insert('myform_data')
  ->fields(array('id','email','first_name'));

$query->values(array(
  'id' => $id,
  'email' => $email,
  'first_name' => $first_name
));

$result = $query->execute();
```

Return TRUE on success and FALSE on failure.

insert into myform_data (id,email,first_name) values (\$id,\$email,\$first_name);



Post Log Messages

```
function watchdog_example(){  
  watchdog('Drupal Camp', 'This is a log entry', NULL, WATCHDOG_DEBUG);  
  $markup = "Check your logs";  
  return array('#markup' => $markup);  
}
```

WATCHDOG_EMERGENCY: Emergency, system is unusable.
WATCHDOG_ALERT: Alert, action must be taken immediately.
WATCHDOG_CRITICAL: Critical conditions.
WATCHDOG_ERROR: Error conditions.
WATCHDOG_WARNING: Warning conditions.
WATCHDOG_NOTICE: (default) Normal but significant conditions.
WATCHDOG_INFO: Informational messages.
WATCHDOG_DEBUG: Debug-level messages.

TYPE	DATE	MESSAGE	USER	OPERATIONS
Drupal Camp	11/14/2014 - 22:05	This is a log entry	emailadmin	



Implementing Javascript

- Drupal uses jQuery!
- But you can use any Javascript code you want
- Lots of ways to implement it - some are better than others
- See <https://www.drupal.org/node/756722>
- Mostly you use `drupal_add_js` function or Drupal Behaviors
- Here are a couple of my examples . . .



Drupal Behaviors

Remember `jQuery(document).ready(function($){ . . . });`; is called when a document is loaded.

- But what about javascript that needs to run after page load and after an AJAX request?
- Drupal Behaviors is:
 - more than `document.ready()`
 - able to run javascript on DOM elements after they appear on a page
- See also <http://www.amazeelabs.com/en/blog/drupal-behaviors-quick-how>



Links

Forms:

- <http://www.sitepoint.com/understanding-forms-drupal/>
- https://api.drupal.org/api/drupal/developer!topics!forms_api_reference.html
- <https://www.drupal.org/node/542202>

