Staying Sane with Drupal

A Developer's Survival Guide

Oscar Merida, @omerida
Drupal GovCon
July 2015

80% of building sites with Drupal is configuration (easy)

Credit: https://www.flickr.com/photos/jeepersmedia/12953571545/

The hard part is the final 20% of customization

- Functionality that is new to Drupal
- or different than assumptions
- Integration with external systems

Credit: https://www.flickr.com/photos/jurvetson/15393495039

Drupal is procedural

- Not Object Oriented
  - Many structures like nodes, forms, users, etc use StdClass objects or arrays
- Not MVC like other PHP frameworks
  - but there is separation of concerns between menu callbacks & theme layers
- Hooks afford a way to alter existing functionality or implement new features.
  - Leverage hooks

Play Well with Others

- Don’t short circuit how things work.
- If you're stuck, ask for help.
- "Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live."
  - http://c2.com/cgi/wiki?CodeForTheMaintainer

Use Existing APIs

- EntityFieldQuery
- Form API
- node_load(), node_presave(), node_* hooks
- Node Access System

Dig into the Documentation

- api.drupal.org has function definitions and examples.

- Security - https://www.drupal.org/writing-secure-code

- Issue Queues

Best Practices

Rule 0: Use VCS

- If you're not already using a Version Control System STOP
    - start using one ASP
- Git, Subversion, etc..
- Rule 0.1 - never edit files on live

Follow Drupal's coding standard.

- Everyone on the team must follow it.
  - https://www.drupal.org/coding-standards
- Don't waste cycles arguing about which standard to follow.
- Can use your IDE to follow a standard.
- BONUS: Use PHP CodeSniffer to enforce it.

Use a Virtual Machine

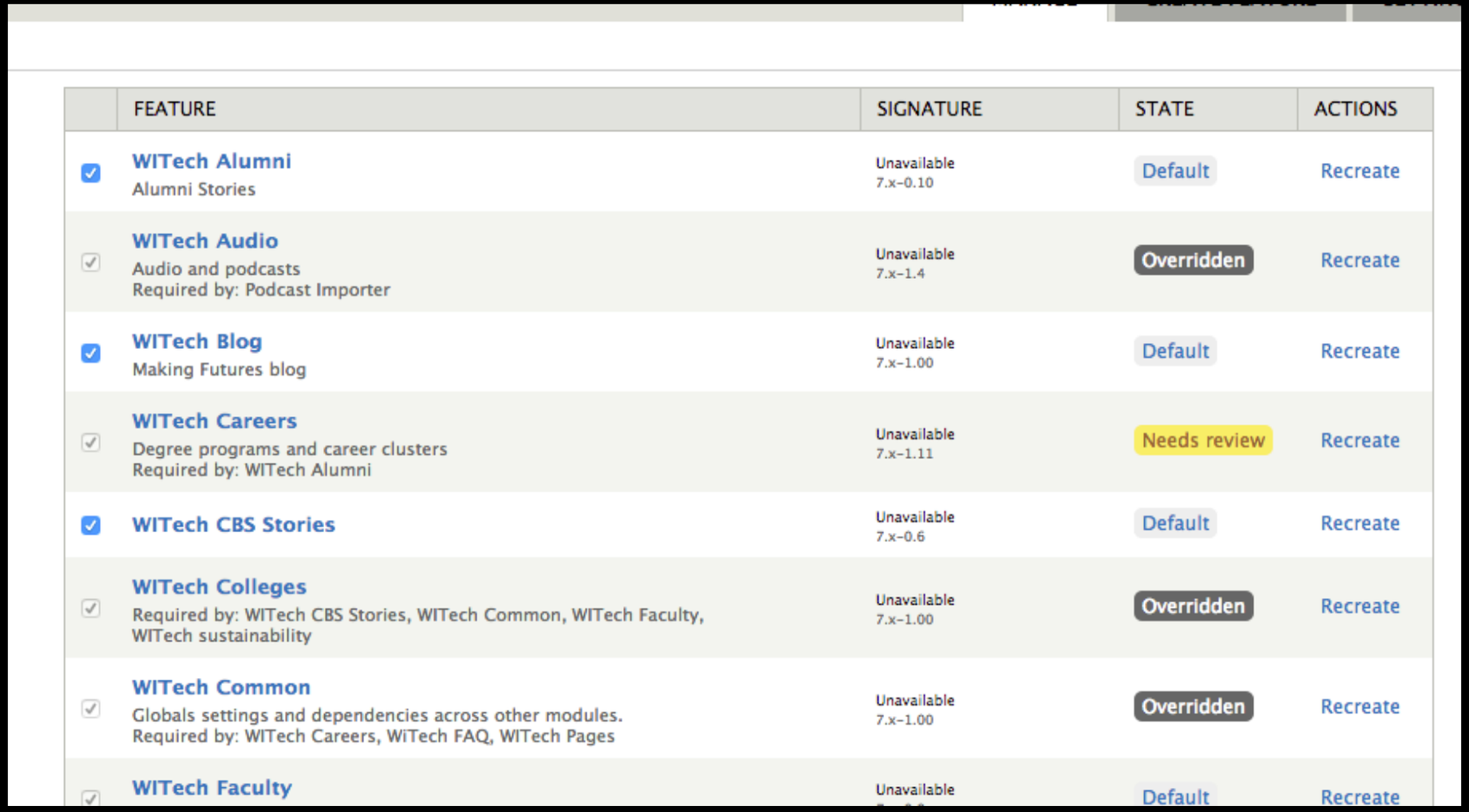
- Use Vagrant to setup a VM for your project
- All developers use the same VM (no more "works on my machine)
- VM should match stage and production environments.
  - Same OS, same PHP, same Mysql, etc.
  - Avoids bugs that only happen on production.

Use Features

- Drupal saves a lot of configuration to the database, making deployments difficult.
- Use Features + Strongarm modules to export settings to Features modules.
  - Create one Feature module for each Content Type
  - Include related fields, Views, Display Suite settings, Panels & Pages or Contexts, Path aliases, Rules,etc.
  - Have a Base module with shared fields and global settings like input formats.

Share Features

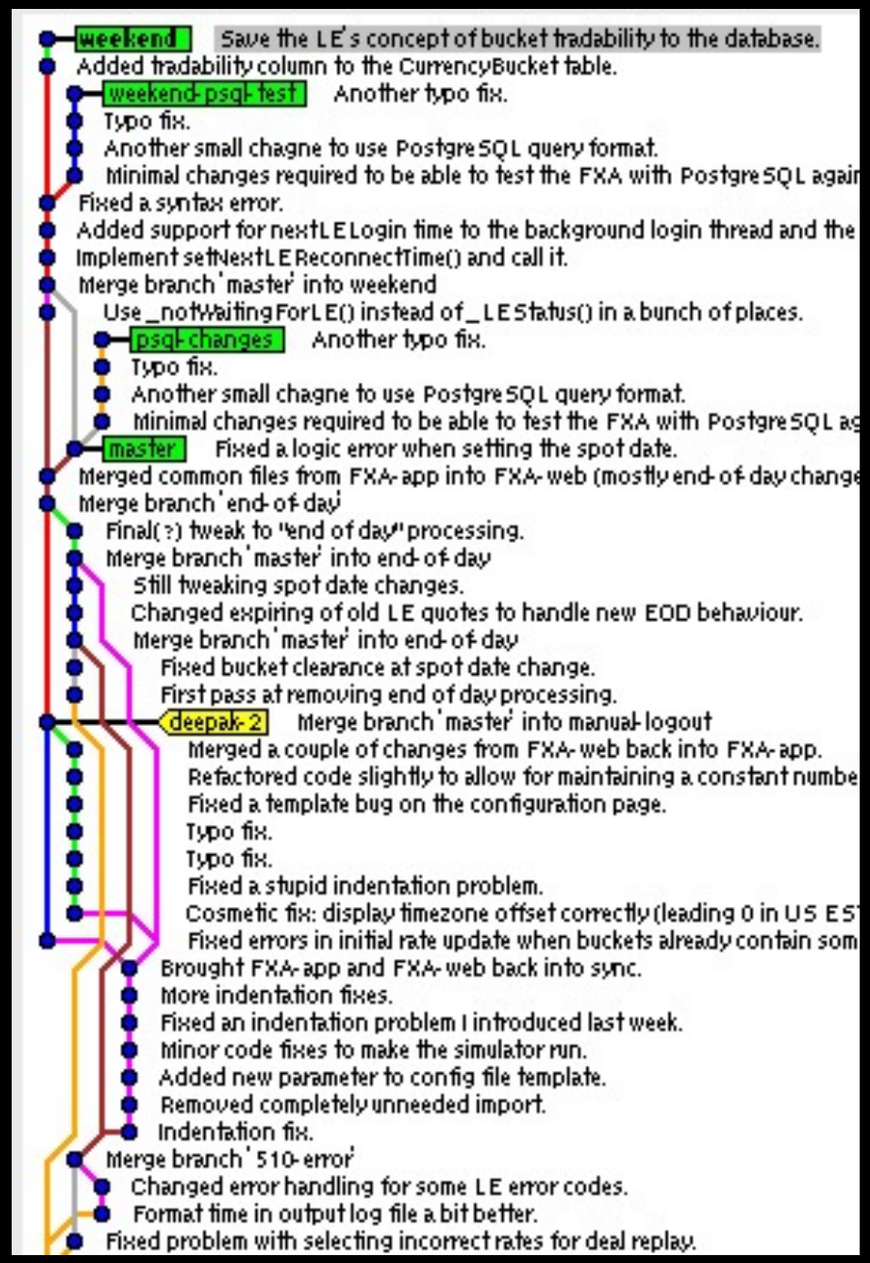
- Track your modules in VCS
- When you update codebase, check Features UI for Overrides and resolve them.



	FEATURE	SIGNATURE	STATE	ACTIONS
☑	**WITech Alumni** Alumni Stories	Unavailable 7.x-0.10	Default	Recreate
☑	**WITech Audio** Audio and podcasts Required by: Podcast Importer	Unavailable 7.x-1.4	Overridden	Recreate
☑	**WITech Blog** Making Futures blog	Unavailable 7.x-1.00	Default	Recreate
☑	**WITech Careers** Degree programs and career clusters Required by: WITech Alumni	Unavailable 7.x-1.11	Needs review	Recreate
☑	**WITech CBS Stories**	Unavailable 7.x-0.6	Default	Recreate
☑	**WITech Colleges** Required by: WITech CBS Stories, WITech Common, WITech Faculty, WITech sustainability	Unavailable 7.x-1.00	Overridden	Recreate
☑	**WITech Common** Globals settings and dependencies across other modules. Required by: WITech Careers, WiTech FAQ, WITech Pages	Unavailable 7.x-1.00	Overridden	Recreate
☑	**WITech Faculty**	Unavailable	Default	Recreate

Use VCS Branches

- Isolate new work on new branches.
- Merge to master/trunk when ready for testing.
- Merge to a stable branch when ready to deploy.
- http://www.mediacurrent.com/blog/git-flow-daily-use



https://www.flickr.com/photos/malcolmtredinnick/1516857444

Automate Deployments

- As simple as having a simple script to sync files to your environments
- Or automagically update dev, stage, production by pushing to the correct VCS branch
  - https://www.freelock.com/node/1108
- BUT remember to keep sensitive information out of VCS (like database credentials).

Don't Fear the Command Line

- Drush is faster and easier to automate than going through the UI.
  - Clearing caches - drush cc all
  - DB Snapshot - drush sql-dump > ../mydb.sql
  - Creating a user - drush user-create
  - Reindexing search - drush search-index

Contrib makes life easy

…except when it doesn’t.

Focus on building new solutions.

- Avoid “Not-invented-here” syndrome
- There's a module for that. In fact, there's probably several …

http://dilbert.com/strip/2014-08-11

But with great power…

- Inherit technical debt of a module
- Keep up-to-date with new releases, especially security releases.
- Sometimes you don't need a module to change things.

Evaluating a module

- Does the maintainer have other modules?
- Does it have a stable release?
- Does it have recent commits?
- Is the issue queue active?
- Does it have good documentation?

Practically core…

- Views (is in D8)
- Views Bulk Operations
- Features & Strongarm
- Pathauto & Token
- Webform
- Rules
- Email
- Link
- Smart Trim
- Redirect
- Entity Reference
- Entity API
- Entity Cache

Nice to Have

- Bean (better Blocks)
- Administration Views
- Display Suite
- Backup & Migrate
- Content Locking
- Menu Block
- Revisioning
- Add Another

Migration tools

- Feeds
- or Migrate
- Pathologic
- Views Data Export

Views

- GUI based Query Builder
- Has its own terminology
- Excels at building display output.
- Not limited to HTML.

https://commons.wikimedia.org/wiki/File:Tower_Optical_Binoculars.jpg

Fields & Filters

- Fields - the fields to return from the query
  - SELECT id, name, birthdate
- Filters - conditions to select items
  - WHERE name=“smith”

Contextual Filters

- Apply filters based on external arguments.
- Filter values usually come from the URL path.
  - Node ID
  - User ID
  - Term ID

Relationships

- Add a related table to the base query
  - JOIN groups ON (groups.id=users.group_id)
- Can then pull in fields from other entites into Fields, Filters, and Contexts.

Use an Existing View Programatically.

- views_embed_view(): Useful for building custom blocks with logic while keeping them configurable in the Views UI.

```
function mymodule_view() {
  $tid = (int) arg(2);
  $preferred = views_embed_view('dept_resources', 'block_1', $tid);

  // test if the view is empty by looking for view-empty class
  if (false !== strpos($preferred, 'class="view-empty"')) {
    $all = views_embed_view('dept_resources', 'block_2', $tid);
    $block['content'] = $all;
  } else {
    $block['content'] = $preferred;
  }
  return $block;
}
```

Views Hooks

- Views API provides a number of hooks for changing a view.
  - hook_views_pre_view()
  - hook_views_pre_ender()
  - hook_views_query_alter()
  - and more! https://api.drupal.org/api/views/views.api.php/group/views_hooks/7

hook_views_pre_view()

- Runs at start of processing. Useful for changing or cleaning up arguments.

```php
function foo_views_pre_view(&$view, &$display_id, &$args) {
  if ('map_legend' == $view->name) {
    // get location short name from aliased view pat
    // ex. "campus/fairfax-va"
    $shortname = parse_path_shortname();
    // get the campus node with the matching shortname
    $node = foo_campus_lookup_shortname($shortname);
    $args[0] = $node->nid;
  }
}
```

hook_views_pre_render()

- Runs at start of rendering.
- Useful for changing titles, adding headers & footers, replacing placeholders.

```php
function foo_views_pre_render(&$view) {
  if ('campus_map' == $view->name) {
    if (isset($view->result[0]->nid)) {
      $node = $view->result[0]->_field_data['nid']['entity'];
      $view->set_title('Campuses for ' . $node->title);
    }
  return;
}
```

hook_views_query_alter()

- Alter a query before it runs.
- Helps deal with input edge cases.

```
function foo_views_query_alter(&$view, &$query) {
  if ('campus_contacts' == $view->name) {
    if ('mid-state' == arg(2)) {
      $query->where[0]['conditions'][0]['value'] = 'Mid-State';
    }
  }
}
```

…but Sometimes

- You need to query for a collection of Entities
- In that case, use EntityFieldQuery
- Unless you're reading & writing your own custom tables, don't use db_query.

```php
function foo_lookup_shortname($short)
{
    // now look up the node for this short to make sure it exists
    $query = new EntityFieldQuery();
    $query->entityCondition('entity_type', 'node')
        ->entityCondition('bundle', 'campus')
        ->propertyCondition('status', 1)
        ->fieldCondition('field_short_name', 'value', $short, '=')
        ->range(0, 1);

    $result = $query->execute();

    if (!$result || empty($result)) {
        return false;
    }

    $ids = array_keys($result['node']);
    $nodes = node_load_multiple($ids);
    $node = array_pop($nodes); // return first
    return $node;
}
```

Writing Custom Modules

Custom modules for:

- Implementing hooks
- Integration with 3rd party systems
- Custom entities

Use Permissions

- hook_perm defines permissions
  - Check for custom permission in routes, views, etc.
  - Avoid checking for one or more roles.

```php
/**
 * Implements hook_permission().
 */
function foo_email_digest_permission() {
  return array(
    'administer foo digest' =>  array(
      'title' => t('Administer FOO Digest'),
      'description' => t('Manage Digest settings.'),
    ),
  );
}
```

Use Node Grants

- hook_node_grants control access to nodes.
- Control user access for viewing, updating, deleting at the node-level.
- Integrates with views and search results.
- https://www.phase2technology.com/drupal-7-node-access-grants-locks-and-keys/

```php
function foo_node_grants($account, $op)
{
  $grants = array();

  // If a user is an admin then grant access to all
  if (user_has_role('Site Admin', $account)) {
    $admin_nids = foo_get_workspace_nids();
    $grants['workspace_member'] = $admin_nids;
  } else {
    $ws_nids = foo_get_users_workspace_nids($account);

    // add the workspace nodes that the user
    // is an owner of too.
    $owner_nids = foo_get_owner_workspace_nids($account);
    $grants['workspace_member']
      = array_merge($ws_nids, $owner_nids);
  }
  return $grants;
}
```

Use Node Grants

- hook_node_grants control access to nodes.
- Control "realms" that have access.
- Integrates with views and search results.
- https://www.phase2technology.com/drupal-7-node-access-grants-locks-and-keys/

```php
function foo_node_grants($account, $op)
{
  $grants = array();

  // If a user is an admin then grant access to all
  if (user_has_role('Site Admin', $account)) {
    $admin_nids = foo_get_workspace_nids();
    $grants['workspace_member'] = $admin_nids;
  } else {
    $ws_nids = foo_get_users_workspace_nids($account);

    // add the workspace nodes that the user
    // is an owner of too.
    $owner_nids = foo_get_owner_workspace_nids($account);
    $grants['workspace_member']
      = array_merge($ws_nids, $owner_nids);
  }
  return $grants;
}
```

Use Node Grants, pt 2

- hook_node_access _records control access to nodes.
- Control user access for viewing, updating, deleting at the node-level.
- Integrates with views and search results.
- https://www.phase2technology.com/drupal-7-node-access-grants-locks-and-keys/

```php
function foo_node_access_records($node)
{
  // don't clutter grants
  if ('resource' !== $node->type) {
    return;
  }

  $ws_nid = $node->field_workspace[LANGUAGE_NONE][0]['target_id'];
  if (empty($ws_nid)) {
    return;
  }

  $grants = array();
  $grants[] = array (
    'realm' => 'workspace_member',
    'gid' => $ws_nid,
    'grant_view' => 1,
    'grant_update' => 0,
    'grant_delete' => 0,
    'priority' => 0,
  );
  return $grants;
}
```

Define custom pages for your module

```php
/**
 * Implements hook_menu().
 */
function foo_email_digest_menu() {
  $items['admin/config/system/foo_digest'] = array(
    'title' => 'FOO Daily Digest',
    'description' => 'Administer FOO Digest settings.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('foo_digest_admin'),
    'access arguments' => array('administer foo digest'),
    'file' => 'foo_email_digest.admin.php',
    'type' => MENU_NORMAL_ITEM,
  );
  return $items;
}
```

Check Requirements

Use hook_requirements to check installation requirements or report status.

```php
/**
 * Implements hook_requirements().
 */
function foo_email_digest_requirements($phase) {
  if ('runtime' == $phase) {
    $toggle = variable_get('foo_digest_toggle', 'off');
    if ('on' == $toggle) {
      $requirements['foo_digest_toggle'] = array(
        'title' => 'FOO Digest Status',
        'value' => check_plain($toggle),
        'severity' => REQUIREMENT_OK,
      );
    } else {
      $requirements['foo_digest_toggle'] = array(
        'title' => 'FOO Digest Status',
        'value' => check_plain($toggle),
        'severity' => REQUIREMENT_WARNING,
      );
    }
```

Expose tasks via Drush

- Make functionality available via drush
  - http://www.drush.org/en/master/commands/
- Can then execute them in Terminal
  - easier to schedule via cron too

```php
/**
 * Implements hook_drush_command().
 */
function foo_email_digest_drush_command() {
  $items['foo-digest-send'] = array(
    'description' => 'Send email digest.',
    'aliases' => array('foodig'),
  );
  return $items;
}
function drush_foo_email_digest_foo_digest_send() {
  foo_digest_send();
}
```

Best Practices

- Create an admin settings form
  - Keep editable settings outside of code
  - Consider having switches to toggle functionality on/off
- Provide theme functions
  - Themes and other modules can override rendered display

Key Global Functions

- t()
- check_plain()
- filter_xss
- arg()
- current_path()
- menu_get_item()
- drupal_goto()
- drupal_deny_access()
- get_current_user()
- drupal_message()

t()

- Output a translatable string with replacements

```php
echo t(
  "Hello @fname",
  array('@fname' => $first_name)
);
```

check_plain()

- Output sanitized text. Important if you're taking user input and displaying it.

```
echo check_plain(
  '<script type="text/javascript">alert("XSS")</script>'
);
```

filter_xss()

- Output sanitized text, **allow some HTML tags**. Important if you're taking user input and displaying it.
- See also filter_admin_xss()

```
echo filter_xss(
  '<strong>Danger ahead!</strong>
  <script type="text/javascript">alert("XSS")</script>'
);
```

arg()

- Get an argument from the URL path. Remember to filter input!

```php
// path is "node/$nid"
// first element is arg(0), second is arg(1), etc...
$nid = (int) arg(1);

if (empty($nid) {
  trigger_error('NID should be an integer', E_USER_ERROR);
}
```

current_path()

- Returns the current **unaliased** path. Useful if a hook should only run on one or more specific pages.

```
if ('node/15' == current_path()) {
    // customize this node
}
```

current_path()

- Returns the current **unaliased** path or the aliased path for an internal one.
- Useful if a hook should only run on one or more specific pages.

```
if ('about' == drupal_get_path_alias()) {
    // customize the 'About Us' page
}

$alias = drupal_get_path_alias('node/15');
```

menu_get_item()

- Returns the currents menu item path. Similar to current_path, easier to test if something should only run on a set of pages.

```
$menu = menu_get_item();
// make sure we're on a user profile page
if ('user/%' !== $menu['path']) {
    return
}
```

drupal_goto()

- Redirect client to another URL. Allows other modules to rewrite paths.

```
if (0 == $user->uid) {
   drupal_goto('user/login');
}
```

drupal_deny_access()

- Show an access denied message.

```
if (0 == $user->uid) {
drupal_deny_access();
                }
```

get_current_user()

```
function foo() {
$user = get_current_user();
```

- Retrieve the user object for the current user.

```
  if (0 == $user->uid) {
drupal_goto(‘user/login’);
                }
                }
```

get_current_user()

function foo() {
er of overwriting or changing $user
- Don't use global $user!

$user->uid = 0;

// rest of function continues
}

drupal_messager()

```
function foo() {
= node_load('node/' . $nid);
 do some stuff to $node
omething went wrong
rupal_set_message(
Could not use node',
ng', // status|warning|error
            );
             }
```

- Display success, warning, or error messages to client (web or Drush).

Thank You. Questions?

- Follow me @omerida
- Will post slides on slideshare.net
- Editor-in-chief of php[architect]
  - Like to write? I want to talk to you!
- Check out http://world.phparch.com