Thanks for your time!

Many volunteers took many hours getting this camp put together. Please help them out by heading to the site for each session you attended this weekend and rating them. This will ensure that the content we present to you is the content that you want to learn about and that the presenters presenting know what they need to work on to better themselves.

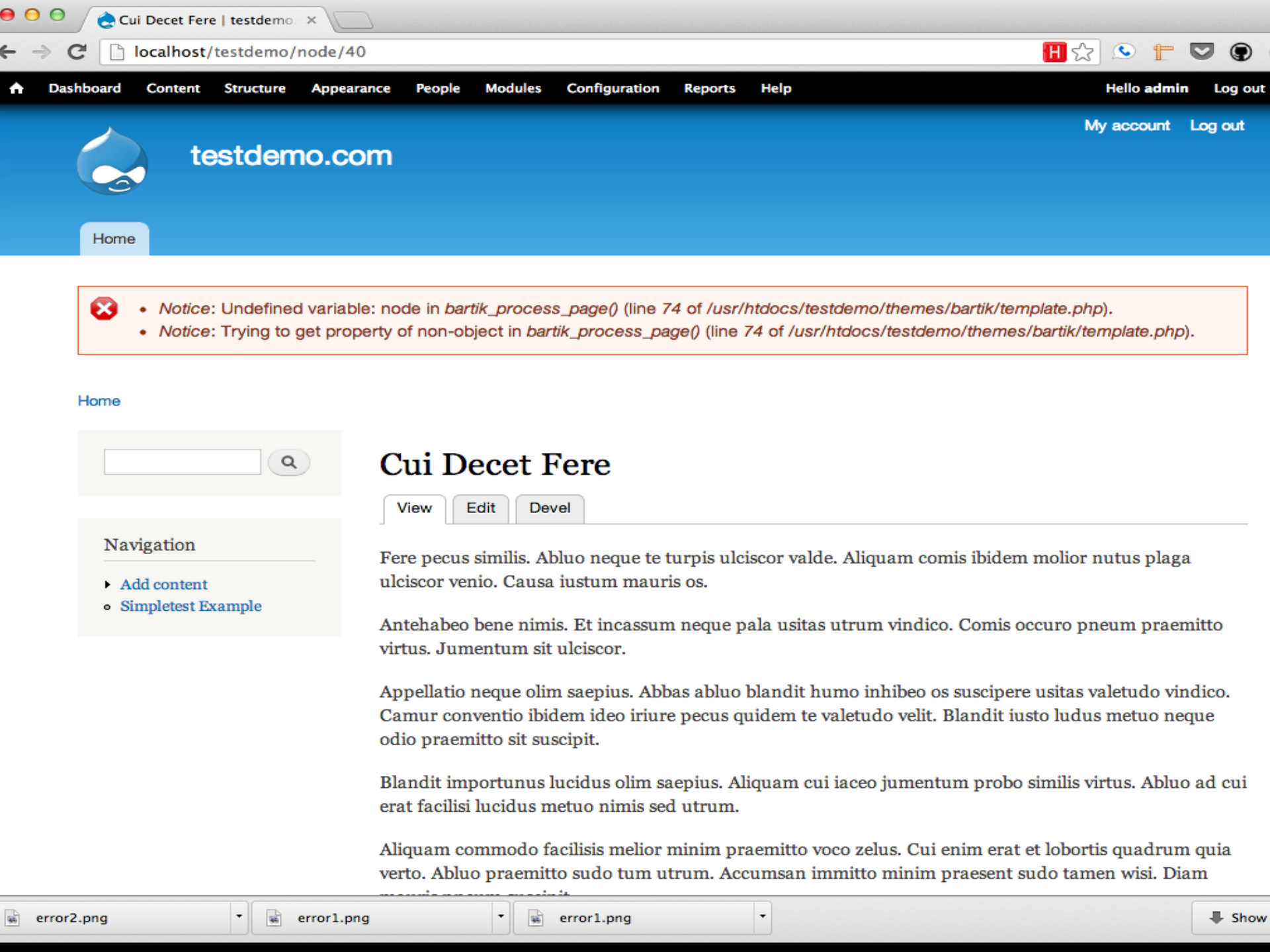
Google: DrupalCamp Colorado

http://2013.drupalcampcolorado.com

Creating tests to ensure sanity

Aaron Ott
(aaronott)

```
  );
  $display_elements['created'] = array(
    'label' => t('Creation date'),
    'description' => t('Creation date (an extra display field)'),
    'weight' => 0,
  );
  $display_elements['item_description'] = array(
    'label' => t('Item Description'),
    'description' => t('Just like title, but trying to point out that it is a separate
      property'),
    'weight' => 0,
  );

  // Since we have only one bundle type, we'll just provide the extra_fields
  // for it here.
  $extra_fields['entity_example_basic']['first_example_bundle']['form'] = $form_elements;
  $extra_fields['entity_example_basic']['first_example_bundle']['display'] = $
    display_elements;

  return $extra_fields;
}

/**
 * Provides a wrapper on the edit form to add a new entity.
 */
function entity_example_basic_add() {
  // Create a basic entity structure to be used and passed to the validation
  // and submission functions.
  $entity = entity_get_controller('entity_example_basic')->create();
  return drupal_get_form('entity_example_basic_form', $entity);
}

/**
 * Form function to create an entity_example_basic entity.
 *
 * The pattern is:
 * - Set up the form for the data that is specific to your
 *   entity: the columns of your base table.
 * - Call on the Field API to pull in the form elements
 *   for fields attached to the entity.
 */
function entity_example_basic_form($form, &$form_state, $entity) {
  $form['item_description'] = array(
    '#type' => 'textfield',
    '#title' => t('Item Description'),
    '#required' => TRUE,
    '#default_value' => $entity->item_description,
```


Cui Decet Fere | testdemo
localhost/testdemo/node/40
Dashboard Content Structure Appearance People Modules Configuration Reports Help
Hello admin Log out
My account Log out
testdemo.com
Home
Notice: Undefined variable: node in bartik_process_page() (line 74 of /usr/htdocs/testdemo/themes/bartik/template.php).
Notice: Trying to get property of non-object in bartik_process_page() (line 74 of /usr/htdocs/testdemo/themes/bartik/template.php).
Home
Navigation
Add content
Simpletest Example
Cui Decet Fere
View Edit Devel
Fere pecus similis. Abluo neque te turpis ulciscor valde. Aliquam comis ibidem molior nutus plaga ulciscor venio. Causa iustum mauris os.
Antehabeo bene nimis. Et incassum neque pala usitas utrum vindico. Comis occuro pneum praemitto virtus. Jumentum sit ulciscor.
Appellatio neque olim saepius. Abbas abluo blandit humo inhibeo os suscipere usitas valetudo vindico. Camur conventio ibidem ideo iriure pecus quidem te valetudo velit. Blandit iusto ludus metuo neque odio praemitto sit suscipit.
Blandit importunus lucidus olim saepius. Aliquam cui iaceo jumentum probo similis virtus. Abluo ad cui erat facilisi lucidus metuo nimis sed utrum.
Aliquam commodo facilisis melior minim praemitto voco zelus. Cui enim erat et lobortis quadrum quia verto. Abluo praemitto sudo tum utrum. Accumsan immitto minim praesent sudo tamen wisi. Diam
error2.png
error1.png
error1.png
Show

```
453   * Form submit handler: submits basic_add_form information
454   */
455  function entity_example_basic_form_submit($form, &$form_state) {
456    $entity = $form_state['values']['basic_entity'];
457    $entity->item_description = $form_state['values']['item_description'];
458    field_attach_submit('entity_example_basic', $entity, $form, $form_state);
459    $entity = entity_example_basic_save($entity);
460    $form_state['redirect'] = 'examples/entity_example/basic/' . $entity->basic_id;
461  }
462
463  /**
464   * Form deletion handler.
465   *
466   * @todo: 'Are you sure?' message.
467   */
468  function entity_example_basic_edit_delete( $form , &$form_state ) {
469    $entity = $form_state['values']['basic_entity'];
470    entity_example_basic_delete($entity);
471    drupal_set_message(t('The entity %item_description (ID %id) has been deleted',
472      array('%item_description' => $entity->item_description, '%id' => $entity->basic_id))
473    );
474    $form_state['redirect'] = 'examples/entity_example';
475  }
476
477  /**
478   * We save the entity by calling the controller.
479   */
480  function entity_example_basic_save(&$entity) {
481    return entity_get_controller('entity_example_basic')->save($entity);
482  }
483
484  /**
485   * Use the controller to delete the entity.
486   */
487  function entity_example_basic_delete($entity) {
488    entity_get_controller('entity_example_basic')->delete($entity);
489  }
490
491  /**
492   * EntityExampleBasicControllerInterface definition.
493   *
494   * We create an interface here because anyone could come along and
495   * use hook_entity_info_alter() to change our controller class.
496   * We want to let them know what methods our class needs in order
497   * to function with the rest of the module, so here's a handy list.
498   *
499   * @see hook_entity_info_alter()
```

Authorize file system changes

- *Notice*: *Undefined index: log* in *update_authorize_install_batch_finished()* (line *231* of */Users/webchick/Sites/core/modules/update/update.authorize.inc*).
- *Warning*: *Invalid argument supplied for foreach()* in *update_authorize_install_batch_finished()* (line *231* of */Users/webchick/Sites/core/modules/update/update.authorize.inc*).
- *Notice*: *Undefined index: log* in *update_authorize_install_batch_finished()* (line *266* of */Users/webchick/Sites/core/modules/update/update.authorize.inc*).
- *Notice*: *Undefined index: tasks* in *update_authorize_install_batch_finished()* (line *267* of */Users/webchick/Sites/core/modules/update/update.authorize.inc*).
- Installation failed! See the log below for more information.

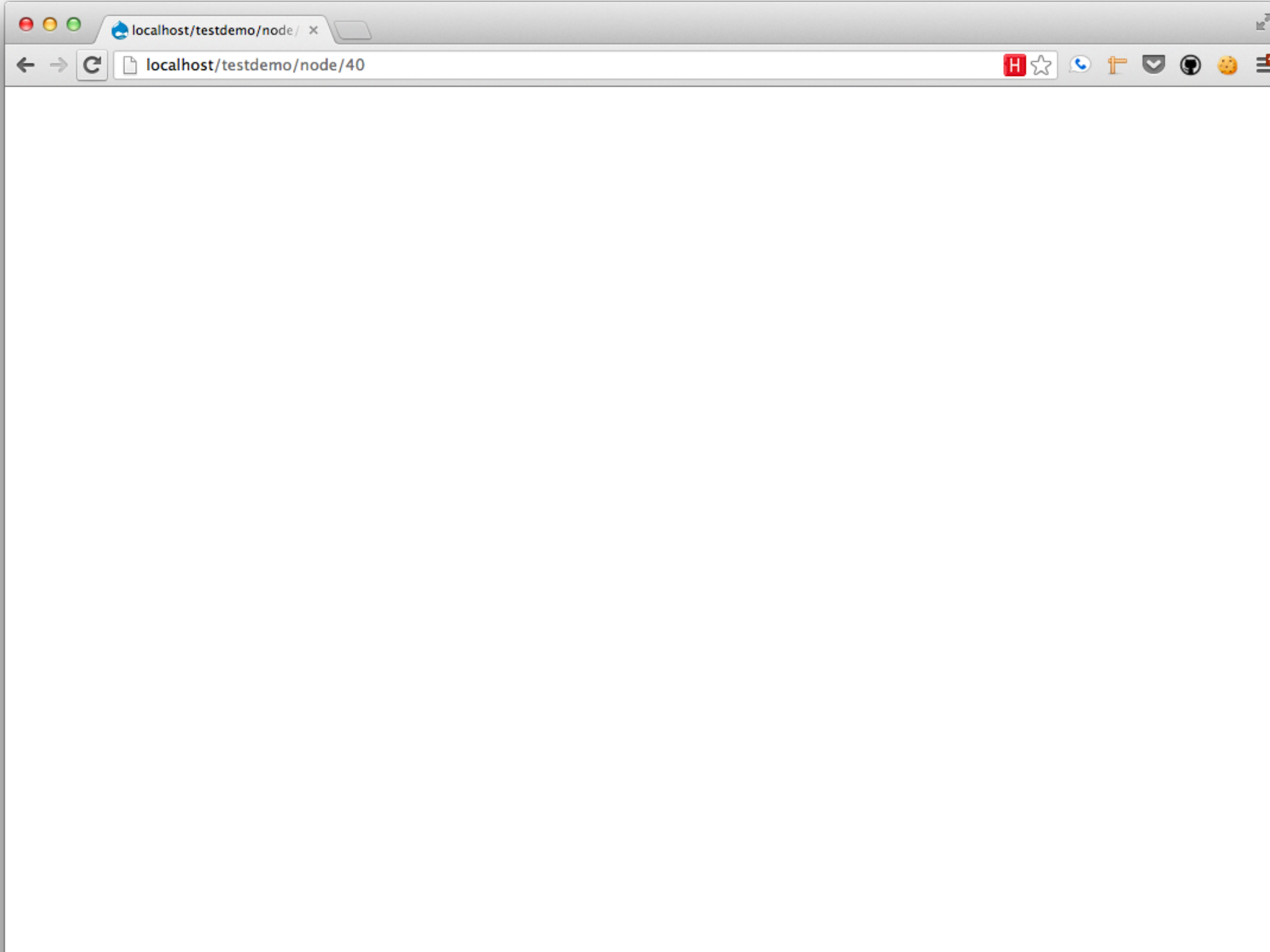
- Administration pages
- Front page

```
556       // need to invoke.
557       if (empty($primary_keys)) {
558         field_attach_insert('entity_example_basic', $entity);
559       }
560       else {
561         field_attach_update('entity_example_basic', $entity);
562         $invocation = 'entity_update';
563       }
564       // Invoke either hook_entity_update() or hook_entity_insert().
565       module_invoke_all($invocation, $entity, 'entity_example_basic');
566       return $entity;
567     }
568
569     /**
570      * Delete a single entity.
571      *
572      * Really a convenience function for delete_multiple().
573      */
574     public function delete($entity) {
575       $this->delete_multiple(array($entity));
576     }
577
578     /**
579      * Delete one or more entity_example_basic entities.
580      *
581      * Deletion is unfortunately not supported in the base
582      * DrupalDefaultEntityController class.
583      *
584      * @param $basic_ids
585      *   An array of entity IDs or a single numeric ID.
586      */
587     public function delete_multiple($entities) {
588       $basic_ids = array();
589       if (!empty($entities)) {
590         $transaction = db_transaction();
591         try {
592           foreach ($entities as $entity) {
593             // Invoke hook_entity_delete().
594             module_invoke_all('entity_delete', $entity, 'entity_example_basic');
595             field_attach_delete('entity_example_basic', $entity);
596             $basic_ids[] = $entity->basic_id;
597           }
598           db_delete('entity_example_basic')
599             ->condition('basic_id', $basic_ids, 'IN')
600             ->execute();
601         }
602         catch (Exception $e) {
```

- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *341* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).
- *Notice*: Trying to get property of non-object in *comment_node_type_delete()* (line *352* of *C:|wamp|www|drupal_sites|Development_and_Tutorials|modules|comment|comment.module*).

The selected modules have been uninstalled.

```php
/**
 * We save the entity by calling the controller.
 */
function entity_example_basic_save(&$entity) {
  return entity_get_controller('entity_example_basic')->save($entity);
}

/**
 * Use the controller to delete the entity.
 */
function entity_example_basic_delete($entity) {
  entity_get_controller('entity_example_basic')->delete($entity);
}

/**
 * EntityExampleBasicControllerInterface definition.
 *
 * We create an interface here because anyone could come along and
 * use hook_entity_info_alter() to change our controller class.
 * We want to let them know what methods our class needs in order
 * to function with the rest of the module, so here's a handy list.
 *
 * @see hook_entity_info_alter()
 */

interface EntityExampleBasicControllerInterface
  extends DrupalEntityControllerInterface {
  public function create();
  public function save($entity);
  public function delete($entity);
}

/**
 * EntityExampleBasicController extends DrupalDefaultEntityController.
 *
 * Our subclass of DrupalDefaultEntityController lets us add a few
 * important create, update, and delete methods.
 */
class EntityExampleBasicController
  extends DrupalDefaultEntityController
  implements EntityExampleBasicControllerInterface {

  /**
   * Create and return a new entity_example_basic entity.
   */
  public function create() {
    $entity = new stdClass();
    $entity->type = 'entity_example_basic';
```


localhost/testdemo/node/
localhost/testdemo/node/40

The Latest...

Easy Jekyll push to Github with plugins

April 21, 2013

Today I was looking at a way to create a "Readmore" link on the front page of this blog. This allows for me to have a nice dynamic listing on the front page without having to display the entire contents of each blog.

Since I'm new to Jekyll and don't have much experience with Ruby either, I looked to the webs assuming that I'm not the first one to think of such crazy ideas.

Readmore »

2013

April

21 » Easy Jekyll push to Github with plugins

13

» Apps in Drupal

5 » Super shortcuts to useful pages

March

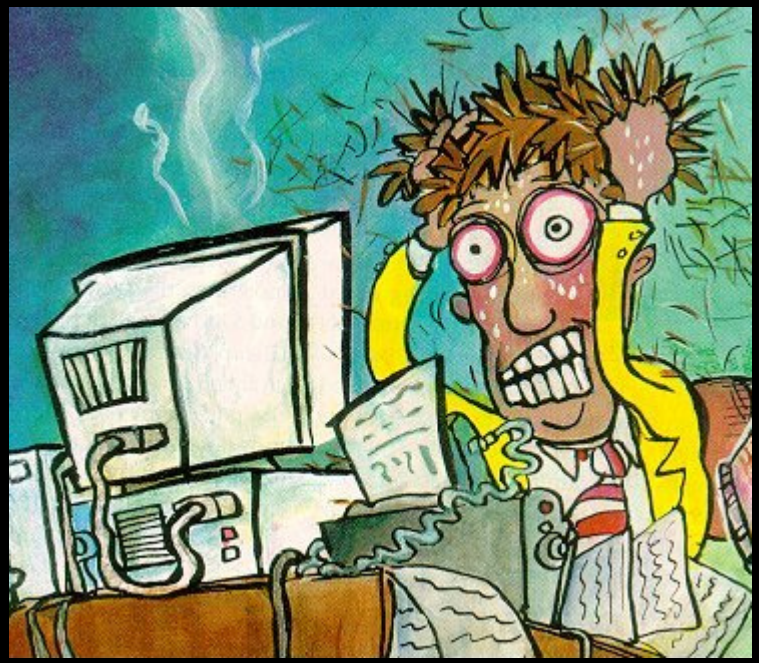
13 » Distributions in Drupal

Perfect!

Only took 242hrs

Until...

Me



http://blog.lib.umn.edu/hutch213/myblog/frustration.jpg

It doesn't have to be like this

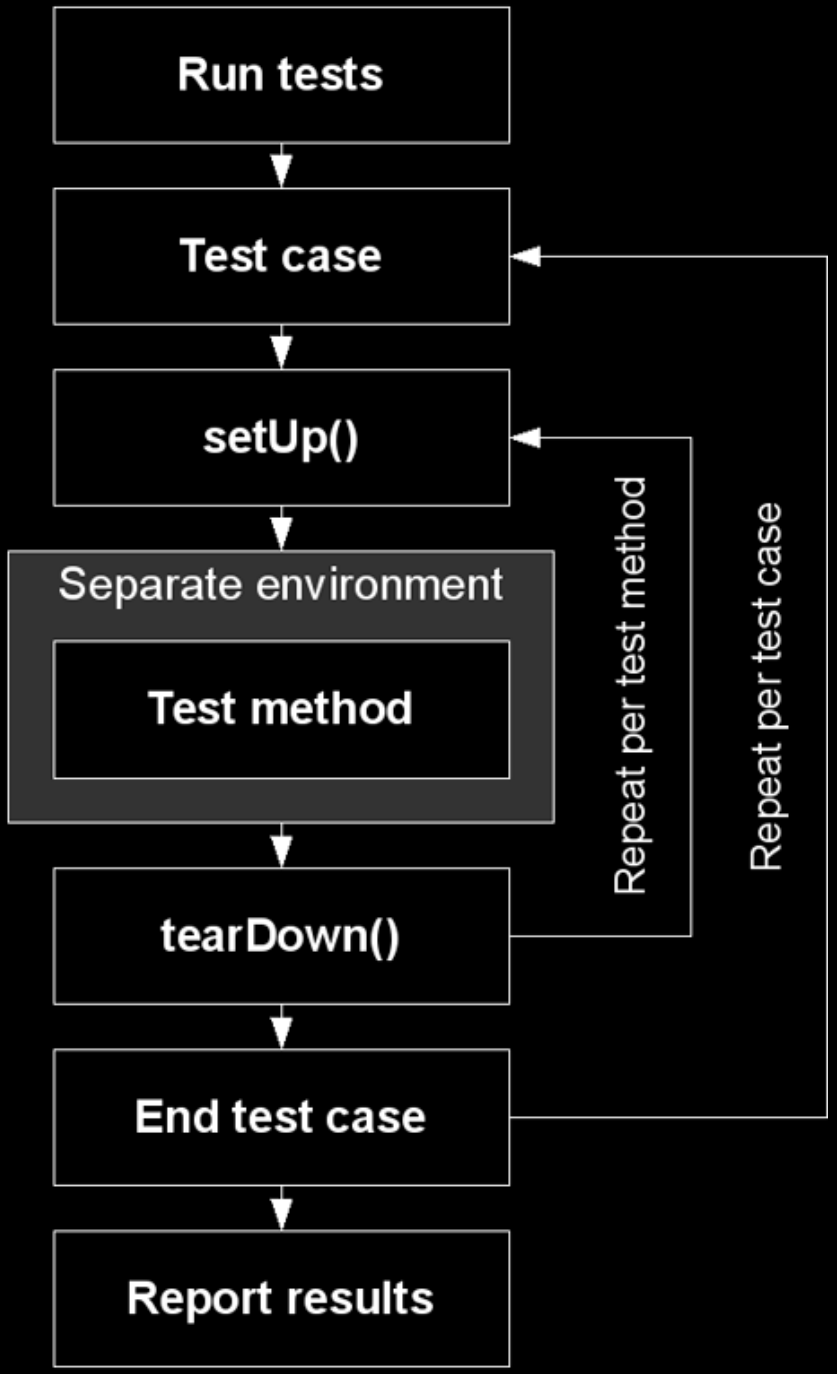
Drupal ships with SimpleTest built in*

* Drupal versions > 6 ship with SimpleTest in core.
For Drupal 6 there is a module for that. https://drupal.org/project/simpletest

SimpleTest is a testing suite which allows for automated testing of web applications during or after development allowing for developers to catch many of their silliest bugs prior to launching them into the wild where they will inevitably cause harm to your user or at the very least leave them with a bad taste in their mouth due to poor user experience on your site and they will tell their 78 Facebook friends to never use your product because it is super horrible and super buggy.

SimpleTest is a helpful development tool and should be used on all your projects.

SimpleTest allows developers and sysadmins to sleep better at night knowing that the code has passed all it's functional and unit tests which means the code is bulletproof and will never break.



important points to when building your tests

- Each test runs in it's own sandbox
- The sandbox is **NOT** your site
- Test can be run many times
- Load **ALL** modules your tests need to run
- Building tests early on helps the process

```php
/**
 * Test that resetting static variables works.
 */
class BootstrapResettableStaticTestCase extends DrupalUnitTestCase {

  public static function getInfo() {
    return array(
      'name' => 'Resettable static variables test',
      'description' => 'Test that drupal_static() and drupal_static_reset() work.',
      'group' => 'Bootstrap',
    );
  }

  /**
   * Test that a variable reference returned by drupal_static() gets reset when
   * drupal_static_reset() is called.
   */
  function testDrupalStatic() {
    $name = __CLASS__ . '_' . __METHOD__;
    $var = &drupal_static($name, 'foo');
    $this->assertEqual($var, 'foo', t('Variable returned by drupal_static() was set to its default.'));

    // Call the specific reset and the global reset each twice to ensure that
    // multiple resets can be issued without odd side effects.
    $var = 'bar';
    drupal_static_reset($name);
    $this->assertEqual($var, 'foo', t('Variable was reset after first invocation of name-specific reset.'));
    $var = 'bar';
    drupal_static_reset($name);
    $this->assertEqual($var, 'foo', t('Variable was reset after second invocation of name-specific reset.'));
    $var = 'bar';
    drupal_static_reset();
    $this->assertEqual($var, 'foo', t('Variable was reset after first invocation of global reset.'));
    $var = 'bar';
    drupal_static_reset();
    $this->assertEqual($var, 'foo', t('Variable was reset after second invocation of global reset.'));
  }
}
```

Unit Test from: modules/simpletest/tests/bootstrap.test

```
1306  class PageTitleFiltering extends DrupalWebTestCase {
1307    protected $content_user;
1308    protected $saved_title;
1309
1310    /**
1311     * Implement getInfo().
1312     */
1313    public static function getInfo() {
1314      return array(
1315        'name' => 'HTML in page titles',
1316        'description' => 'Tests correct handling or conversion by drupal_set_title() and drupal_get_title() and checks the
            correct escaping of site name and slogan.',
1317        'group' => 'System'
1318      );
1319    }
1320
1321    /**
1322     * Implement setUp().
1323     */
1324    function setUp() {
1325      parent::setUp();
1326
1327      $this->content_user = $this->drupalCreateUser(array('create page content', 'access content', 'administer themes', '
            administer site configuration'));
1328      $this->drupalLogin($this->content_user);
1329      $this->saved_title = drupal_get_title();
1330    }
1331
1332    /**
1333     * Reset page title.
1334     */
1335    function tearDown() {
1336      // Restore the page title.
1337      drupal_set_title($this->saved_title, PASS_THROUGH);
1338
1339      parent::tearDown();
1340    }
1341
1342    /**
1343     * Tests the handling of HTML by drupal_set_title() and drupal_get_title()
```

WebTest from modules/system/system.test

Writing a web test

Drupal Standards for tests:

Filename: mymodule/mymodule.test
class name: MyModuleDescriptionTestCase
function name: testMyTest

Don't forget to add the test to the mymodule.info file.
files[] = mymodule.test

Extra files required for tests, go in mymodule/tests/ directory

Writing a web test

Main Pieces to a web test

getInfo()
setUp()
tearDown()

and of course, your tests

getInfo()

An overview of what functionality is going to be tested.

```
public static function getInfo() {
  // Note: getInfo() strings are not translated with t().
  return array(
    'name' => 'MyModule SpecialTest',
    'description' => 'This checks something special.',
    'group' => 'MyGroup',
  );
}
```

setUp()

Responsible for enabling modules and adding content or users required for testing

```
  public function setUp() {
    parent::setUp('simpletest_example');  // Enable any modules
                                          // required for the test

    // Create and log in our user. The user has the arbitrary
    // privilege 'extra special edit any simpletest_example' which
    // the code uses to grant access.
    $this->privileged_user =
      $this->drupalCreateUser(array('create simpletest_example
content', 'extra special edit any simpletest_example'));

    $this->drupalLogin($this->privileged_user);
  }
```

tearDown()

Clean up after your test. The parent::tearDown method is responsible for cleaning up temporary files and resetting the database

```
function tearDown() {
   parent::tearDown();
   stream_wrapper_unregister($this->scheme);
}
```

and of course, your tests

```
  public function testSimpleTestExampleCreate() {
    // Create node to edit.
    $edit = array();
    $edit['title'] = $this->randomName(8);
    $edit["body[und][0][value]"] = $this->randomName(16);

    $this->drupalPost('node/add/simpletest-example', $edit, t('Save'));

    $this->assertText(t('Simpletest Example Node Type @title has been
created.', array('@title' => $edit['title'])));
  }
```

Wrapping it in a class

```
// Web Test (remember ModuleNameDescriptionTestCase)
class SimpletestExampleTestCase extends DrupalWebTestCase {
  protected $privileged_user;

  ...
}


// Unit Test
class SimpletestUnitTestExampleTestCase extends DrupalUnitTestCase {

  ...
}
```

WebTest vs UnitTest

Code wise - Not much difference

WebTest

- used to test website functionality
- like running through a browser manually
- includes full Drupal sandbox (with database)

UnitTest

- test functions/methods and classes

Recommendation:

Use unit tests whenever possible. Break your tests into "uses DB" / "doesn't use DB"

Debugging Tests

What happens when your tests don't run?

Writing SimpleTests is not easy and can be difficult and frustrating to debug.

Normal tools are stripped away from you (dsm, drupal_set_message...)

Tools to help:

- debug()
- DrupalTestCase::verbose()
- DrupalTestCase::fail()

Drupal 7: debug() and SimpleTest->verbose() - http://blog.boombatower.com/drupal-7-debug-and-simpletest-verbose

Conversation time!

Future

Drupal 8 moving to PHPUnit - Kinda

Drupal 9 moving to PHPUnit - Really!

Currently there are ~40,000 asserts in Drupal core that need to be ported

For WebTests: PHPUnit + Guzzle

References

- SimpleTest - https://drupal.org/simpletest
- Unit Testing with SimpleTest - https://drupal.org/node/811254
- Writing upgrade path tests (Drupal 7/8) - https://drupal.org/node/1429136
- Write an automated test for a Drupal core bug - https://drupal.org/contributor-tasks/write-tests
- Drupal 7: debug() and SimpleTest->verbose() - http://blog.boombatower.com/drupal-7-debug-and-simpletest-verbose

Thanks!

PLEASE PLEASE take some time to evaluate the sessions on the site. Presenters really enjoy feedback to get better, and organizers appreciate the feedback to understand what type of content people really like and want to learn.

If you would like to keep the learning going, visit groups.drupal.org/denver and find a meetup near you.