

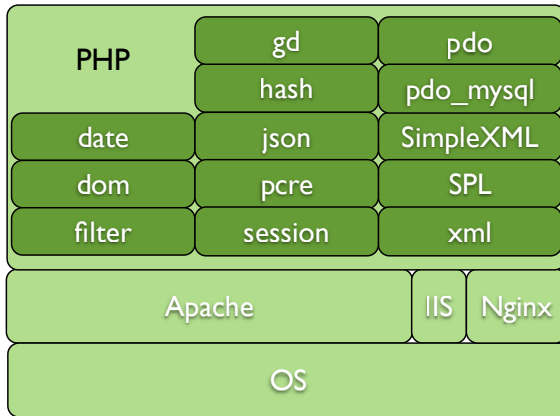
# An Introduction to Drupal Architecture

John VanDyk  
DrupalCamp Des Moines, Iowa  
September 17, 2011



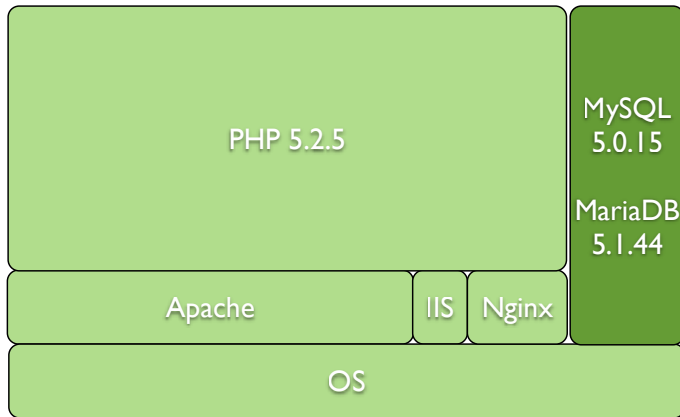
2

Stack with OS, webserver and PHP. Most people use mod\_php but deployments with engineers at the helm sometimes use FastCGI. PHP 5.2.4 with the 5.2.5 security backport is fine too (Ubuntu 8.0.4).



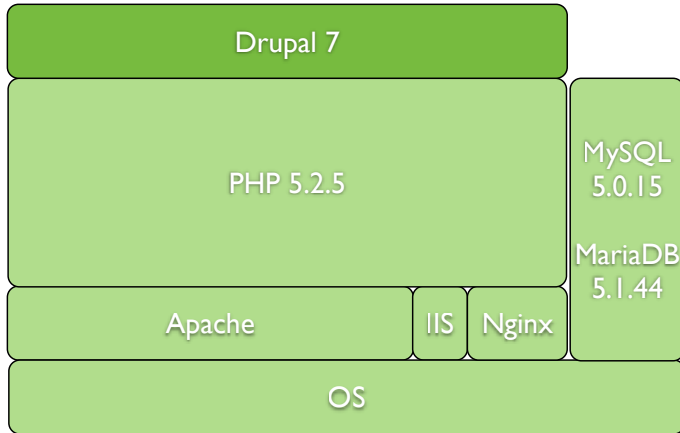
3

Some PHP extensions are required.



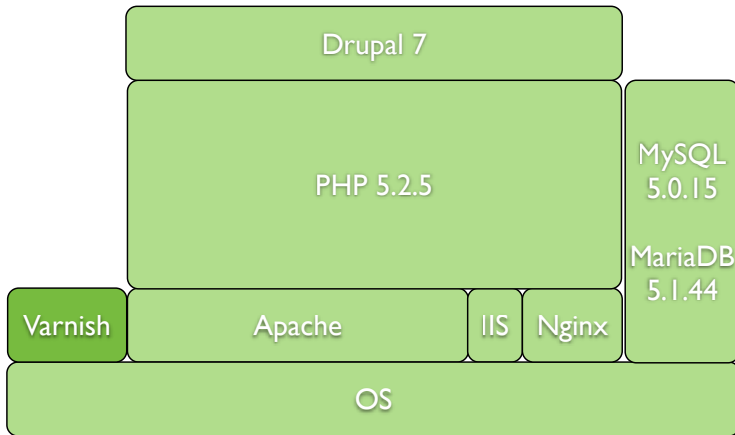
4

Database.



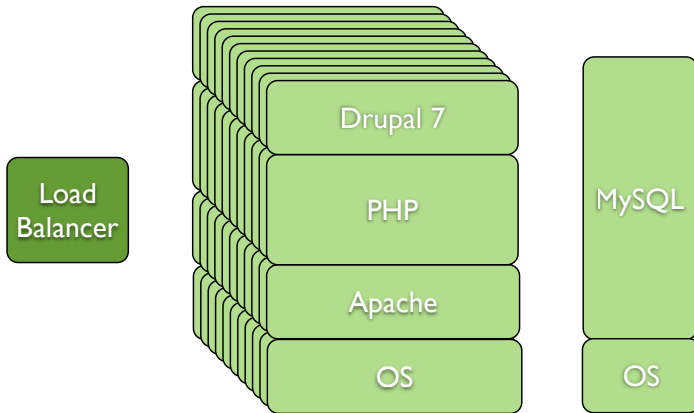
5

Drupal sits atop the stack.



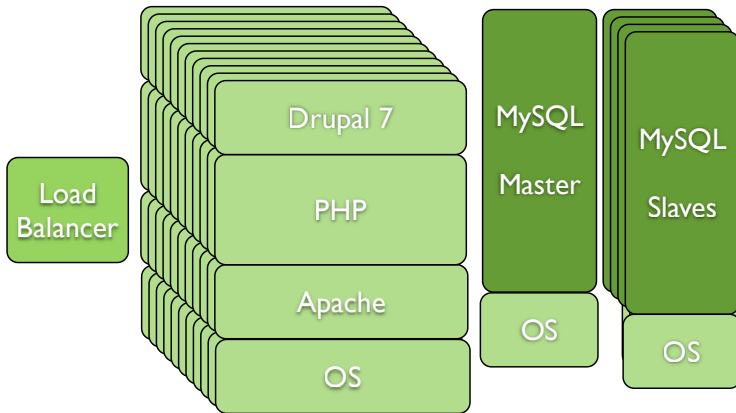
6

Varnish can be added to the front end for fast RAM-based response to anonymous users.



7

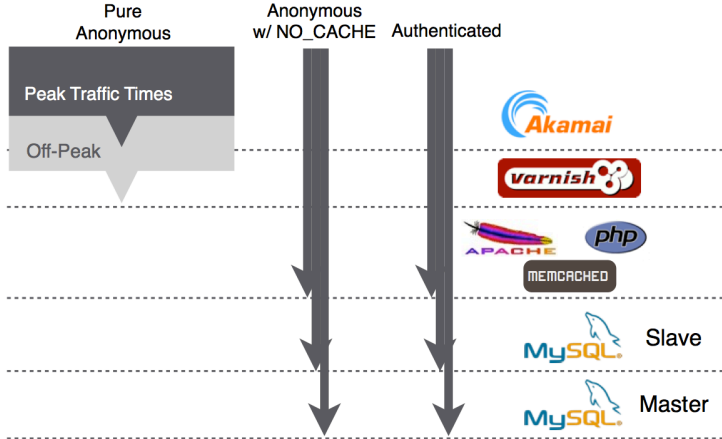
Webheads can be scaled out.



8

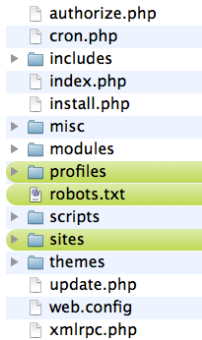
Databases can be scaled out. Show master-slave but master-master is also a possibility. See <http://www.jochus.be/site/2011-01-13/drupal/drupal-master-master-replication-architecture> for an example.





Source: Nate Haug, Lullabot *Scaling the Grammys*, DrupalCamp Denver 2010.

214 million hits in six hours at peak. Database load: 5%. Apache load: 1%.

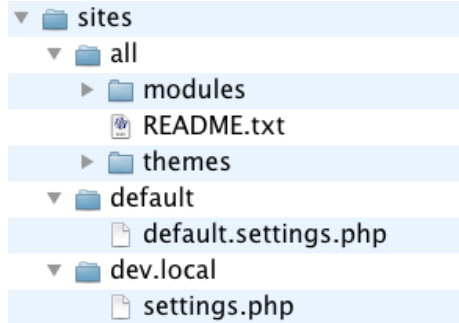


**You will not hack core.**

10

You should only modify the locations shown in green. Modifying the other files will inevitably lead to grief.

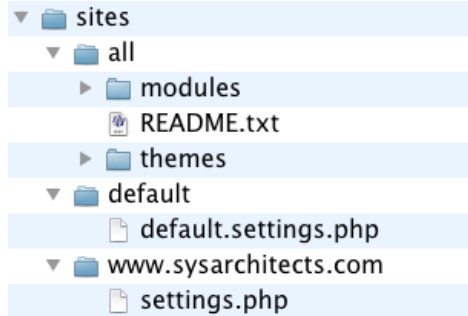
# Development Site



11

Local settings, including database connection info, are in sites/all/dev.local.

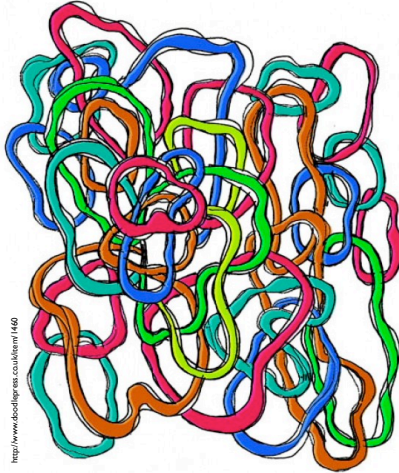
# Live Site



12

On the live site settings are in a site-specific directory.

## Drupal 7 Architecture



13

Drupal architecture. Each include file has a different color and their interrelationships are shown. Uh...just kidding.

Browser request: GET /node/1

Bootstrap (includes/bootstrap.inc)

- brings session, database, variables online

Menu System (router)

- maps URL to a function
- checks access control
- may bring functions into scope
- calls function to build page data structure
- calls function to theme page (look and feel)
- page returned to client

It is important for a developer to understand the sequence of events that occur when a request is processed by Drupal.

/node/1



mod\_rewrite in .htaccess

/index.php?q=node/1

`$_GET['q']` will contain 'node/1'

15

A typical request starts with mod\_rewrite.

index.php

menu system

Which function should this path be mapped to?

```
$items['node/%node'] = array(  
  'page callback' => 'node_page_view',  
  'page arguments' => array(1),  
  'access callback' => 'node_access',  
  'access arguments' => array('view', 1),  
);
```

16

index.php invokes the menu system, which handles much more than menus. The menu item that maps to node/1 can be found in node.module.



index.php

menu system

Does this user have permission to access this path?

```
$items['node/%node'] = array(  
  'page callback' => 'node_page_view',  
  'page arguments' => array(1),  
  'access callback' => 'node_access',  
  'access arguments' => array('view', 1),  
);
```

17

node.module tells the menu system to invoke a function (node\_access()) to determine whether the current user has permission.

index.php

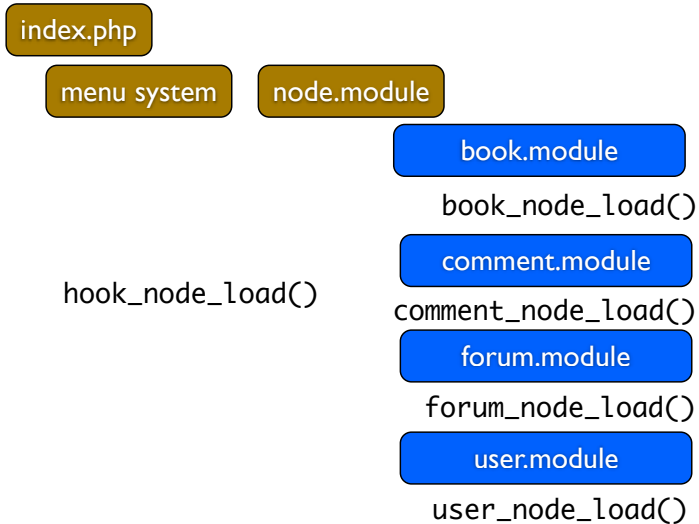
menu system

node.module

```
/**
 * Menu callback; view a single node.
 */
function node_page_view($node) {
  ...
}
```

18

The page callback for the path node/1 is 'node\_page\_view'. That's the name of the function that is called if the user has permission to view the page.



19

Other modules have a say in the loading of the node object. Many of these hooks (better thought of as "events") happen during the building of a page.

index.php

menu system

```
nodes = Array [2]
  #sorted = (boolean) true
  1 = Array [11]
    #bundle = (string:4) page
    #entity_type = (string:4) node
    #language = (string:2) en
    #node = Object of: stdClass
    #pre_render = Array [1]
    #theme = (string:4) node
    #view_mode = (string:4) full
    #weight = (int) 0
    body = Array [16]
      #access = (boolean) true
      #bundle = (string:4) page
      #entity_type = (string:4) node
      #field_name = (string:4) body
      #field_translatable = (string:1) 0
      #field_type = (string:17) text_with_summary
      #formatter = (string:12) text_default
      #items = Array [1]
      #label_display = (string:6) hidden
      #language = (string:3) und
      #object = Object of: stdClass
      #theme = (string:5) field
      #title = (string:4) Body
      #view_mode = (string:4) full
      #weight = (int) 0
      0 = Array [1]
        #markup = (string:11) <p>Hi.</p>
    comments = Array [0]
    links = Array [5]
```

20

A large array with the node's data structure is returned to the menu system.

index.php

menu system

node.module

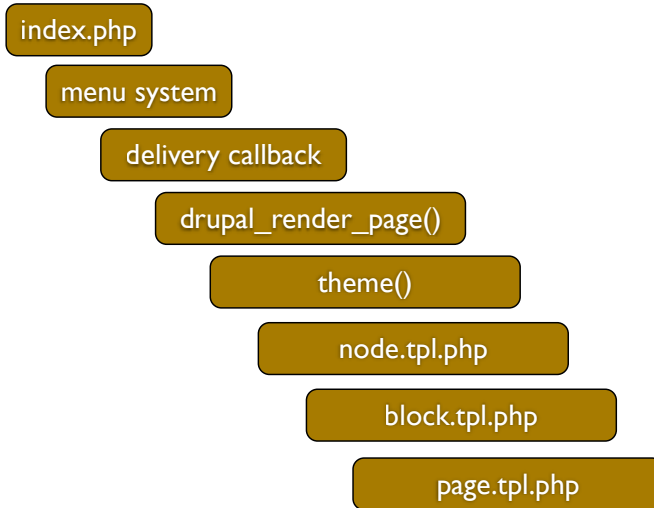
hook\_node\_view\_alter()

drupalcorn.module

drupalcorn\_node\_view\_alter()

21

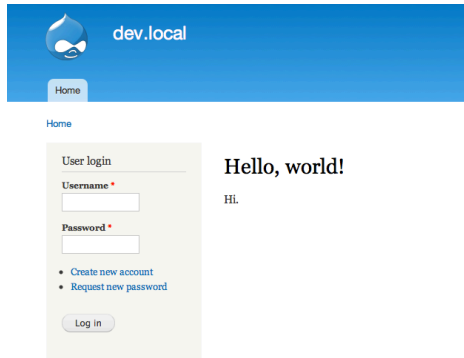
Other modules may alter the final data structure. We've seen three common patterns that pervade Drupal: (1) hooks allow other modules to respond to events; (2) Drupal uses large structured arrays, and (3) other modules may alter these arrays once built.



22

The menu system passes the structure array to a delivery callback which turns the array into HTML by rendering it; that is, passing it through Drupal's template engine. Template files (.tpl.php, pronounced "tippie-fipp") in the current theme are used to create the HTML.

index.php



The screenshot shows a web application interface. At the top is a blue header bar with a logo on the left and the text "dev.local" on the right. Below the header is a navigation bar with a "Home" button. The main content area is divided into two sections. On the left is a "User login" form with fields for "Username" and "Password", each with a red asterisk indicating a required field. Below the password field are two links: "Create new account" and "Request new password". At the bottom of the form is a "Log in" button. On the right is a large text area displaying "Hello, world!" in a bold font, with "Hi." in a smaller font below it.

dev.local

Home

Home

User login

Username \*

Password \*

- [Create new account](#)
- [Request new password](#)

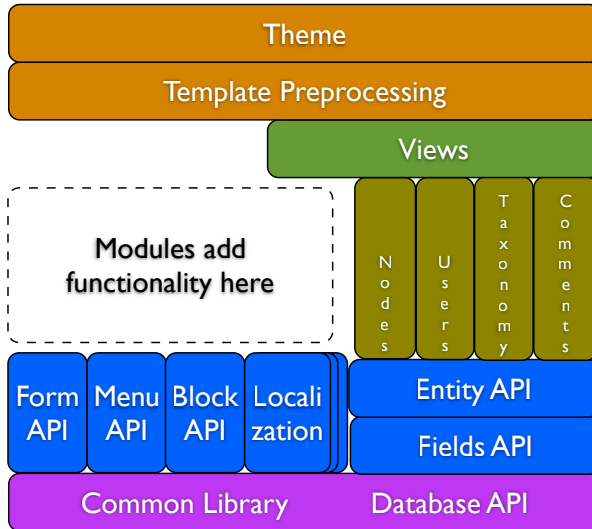
Log in

**Hello, world!**

Hi.

23

The HTML is then returned to the browser.



24

Block diagram if you need a mental model. This doesn't have nearly enough dimensions to describe what actually happens. If you asked 5 developers to make Drupal block diagrams, you'd get 5 different diagrams. Hopefully this one helps someone.



## "Entities"

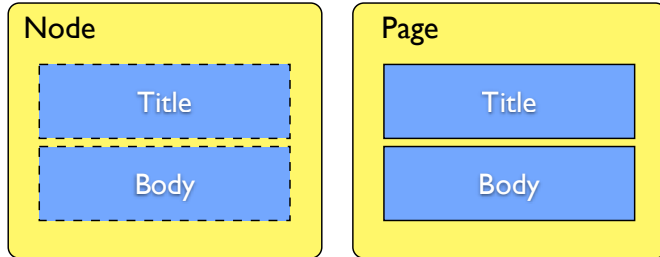
Node: a piece of content

User: a user account

Taxonomy: categories for tagging/classification

Comment: content with a parent

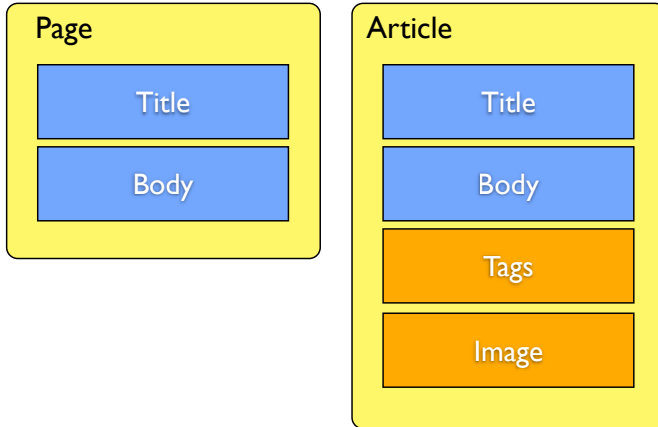
# An Entity has Bundles



26

An entity can be thought of as a base object, and bundles are subclasses. For example, a node is an entity while the bundle of fields that make up a page are a type of node.

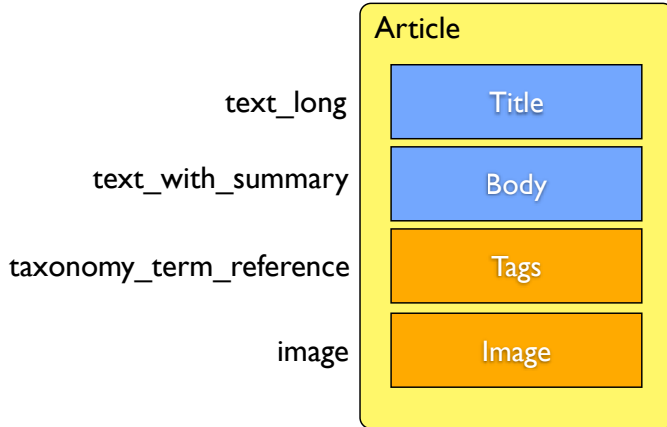
# An Entity has Bundles



27

These are types of nodes. Each has a bundle of fields.

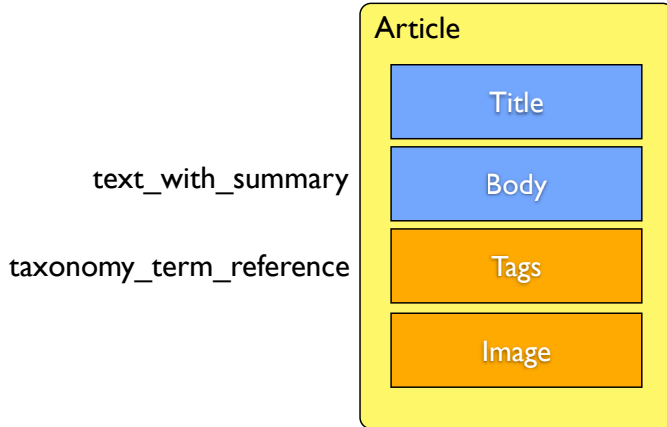
# A Bundle has Fields



28

And each field has a type.

# Not SQL Fields



29

These are more than SQL fields. `text_with_summary` autogenerates a summary. `taxonomy_term_reference` splits multiple tags into separate records. These are Drupal fields.

# In the Real World

"Entity" = node | user | comment | taxonomy

"Bundle" = "Content Type"

You assemble content types from fields using the UI.

Part of the conversation with your client!

30

Nobody says "let's create a new Node Bundle and call it Bug Report".

# Before Building a Site

- What content types and fields?
- How will the data get in? (Forms)
- How will the data get out? (Views)
- Who may create/view/delete data? (Permissions)
- What additional functionality? (Modules)
- How can we make it pretty? (Theming)

31

Questions to map client needs to Drupal concepts.

## Documentation

[Docs Home](#) | [API](#) | [Recently Updated](#)

### API reference

[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 "Components of Drupal" section, 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](#)

#### Search 7

Function, file, or topic: \*

#### API Navigation

Drupal 7

[Constants](#)

[Classes](#)

[Files](#)

[Functions](#)

[Globals](#)

[Topics](#)

<http://api.drupal.org>

32

Your home for API-related searches.



<http://drupal.org/project/examples>



The screenshot shows the Drupal.org website interface. At the top is a blue header with the Drupal logo and a search bar. Below the header is a navigation bar with links: Drupal Homepage, Your Dashboard, Logged in as jvandyk, Log out, and Admin. The main content area has a section titled "Download & Extend" with sub-links: Download & Extend Home, Drupal Core, Modules, Themes, Translations, and Installation. Below this is a section titled "Examples for Developers" with a "View" button and links for Version control, Edit, Outline, and Revisions. The text below states: "Posted by rfay on October 3, 2009 at 2:03pm". It then describes the project's aim to provide high-quality, well-documented API examples for a broad range of Drupal core functionality. A link is provided for those interested in other, non-core examples. A paragraph explains that developers can learn how to use a particular API quickly by experimenting with the examples, and adapt them for their own use. Finally, a "News" section lists several updates: 13 Dec 2010: Action Example added to D7 Examples; 14 Oct 2010: An extensible multistep form example added to D7 Form Example; 14 Oct 2010: A Drupal 6 Token Example; 2 Oct 2010: Major rework and improvement of XMLRPC Example; 26 Sept 2010: Added a Render Example showing how the D7 Render subsystem works; 5 Sept 2010: Element Example improved (new examples), moved into Form Example, and text added.

Drupal™

Drupal Homepage Your Dashboard Logged in as jvandyk Log out Admin

Download & Extend

Download & Extend Home Drupal Core Modules Themes Translations Installation P

### Examples for Developers

[View](#) [Version control](#) [Edit](#) [Outline](#) [Revisions](#)

Posted by [rfay](#) on October 3, 2009 at 2:03pm

This project aims to provide high-quality, well-documented API examples for a broad range of Drupal core functionality.

(Interested in [other, non-core examples?](#))

Developers can learn how to use a particular API quickly by experimenting with the examples, and adapt them for their own use.

#### News

- 13 Dec 2010: Action Example added to D7 Examples
- 14 Oct 2010: An extensible multistep form example added to D7 Form Example
- 14 Oct 2010: A Drupal 6 Token Example
- 2 Oct 2010: Major rework and improvement of XMLRPC Example
- 26 Sept 2010: Added a Render Example showing how the D7 Render subsystem works
- 5 Sept 2010: Element Example improved (new examples), moved into Form Example, and text added

33

Simple implementations of APIs. You can base your own work on these examples.

# Bonus!

Two tools I use all the time.

# Defeating the White Screen of Death

Define PHP error log in php.ini:

```
error_log = /var/log/php/error.log
```

35

I close with two essential tools. First, don't be frightened by the WSOD.

Watch the error log to find out what's going on:

```
tail -f /var/log/php/error.log
```

```
[07-Sep-2011 16:25:02] PHP Fatal error:  
Call to undefined function int() in  
bgstat.module on line 41
```

36

PHP will always tell you what went wrong in its log. If you've defined a PHP error log in php.ini, you can watch it in real time with tail -f.

# Where is the code?

Quickly search the codebase for a function:

```
egrep -rn searchstring .
```

37

Second, don't be overwhelmed with the codebase. `egrep` and pipes can help you find where functions live and where things happen. Example: where are mail-related variables set?

```
egrep -rn mail . | grep variable_set

./includes/install.core.inc:1802:
variable_set('site_mail', $form_state['values']
['site_mail']);

./includes/install.core.inc:1813:
variable_set('update_notify_emails', array($form_state
['values']['account']['mail']));

./modules/openid/openid.test:323:
variable_set('user_email_verification', TRUE);
...
```

38

There we go. The codebase is like a rich porridge, waiting for you to sift through and find the tasty lumps.

Thanks!

Hopefully these help create a big picture  
of how Drupal works. This only scratches the surface.

Now get your debugger  
working and explore Drupal core on your own!