

Table of Contents

| | |
|---|----|
| Drupal handbook | 1 |
| About Drupal | 2 |
| Is Drupal right for you? | 2 |
| Gallery | 3 |
| acko.net | 3 |
| Cam Oca&Ä± | 3 |
| Can't Eat It | 3 |
| Copenhagen | 4 |
| Drupal Hungary | 4 |
| Echo Ditto | 4 |
| Eric Scouten Photography | 4 |
| Footnotes | 5 |
| Fulbright - EspaÃ±a | 5 |
| Hip Foto | 5 |
| Kernel Trap | 6 |
| Our Media | 6 |
| Pop Madrid | 6 |
| Snowboard Magazine | 7 |
| Spread Firefox | 7 |
| Terminus 1525 | 7 |
| URLGreyHot | 8 |
| Case studies | 8 |
| Sites that use Drupal | 9 |
| Success stories | 9 |
| Contaire.com: a corporate website based on Drupal | 9 |
| Our requirements | 9 |
| The ingredients | 10 |
| A dynamic horizontal tab menu | 10 |
| A new teaser.module | 11 |
| Two columns, but sorted, please | 12 |
| Conclusion | 12 |
| Why Linux Journal converted to Drupal and how it went | 12 |
| Feature overview | 14 |
| Typical Drupal applications | 14 |
| Rolling your own system vs. using Drupal | 15 |
| The Drupal path is fun and educational | 15 |
| Drupal's user and developer base | 15 |
| With Drupal, you are able to create a cool site | 16 |
| Drupal is flexible | 16 |
| You can grow in the future with Drupal | 17 |
| Background | 17 |
| History | 17 |
| Druplicon (the logo) | 18 |
| Mission | 18 |

| | |
|--|----|
| Principles | 18 |
| Usability aims | 19 |
| Presentations and articles | 19 |
| Community | 20 |
| Development and support | 20 |
| The Drupal core | 20 |
| Contributions | 21 |
| Documentation and support | 21 |
| Users | 21 |
| Download | 21 |
| Support | 22 |
| Hosting and services | 22 |
| Drupal hosting | 22 |
| GrafiX Internet B.V. | 22 |
| Norrix | 23 |
| OpenSourceHost | 23 |
| CascadeHosting | 23 |
| Drupal services | 24 |
| Moshe Weitzman | 24 |
| Teledynamics Communications | 25 |
| Gerhard Killesreiter | 25 |
| Steven Wittens | 26 |
| Webschuur.com | 26 |
| Trae McCombs and Kyle Smith | 27 |
| Heydon Consulting - Gordon Heydon | 27 |
| Matt Westgate | 28 |
| Károly Négyesi | 28 |
| 2bits.com (Canada) | 29 |
| Donating to the Drupal project | 29 |
| Donate | 29 |
| Expenses | 29 |
| Donors | 30 |
| Newsletter advertising | 33 |
| Version 4.6 roadmap | 34 |
| User's guide | 37 |
| Registering as a user | 37 |
| Logging in | 38 |
| Changing your account settings | 38 |
| Creating new content | 39 |
| A step-by-step example | 39 |
| Types of content | 41 |
| Topics, categories and terms | 41 |
| Permissions | 41 |
| Moderation and the submission queue | 41 |
| Creating comments | 42 |
| Alternative ways to enter content | 42 |
| Posting and editing content with w.bloggar | 43 |

| | |
|--|----|
| Preparing content offline | 43 |
| Editing and deleting content | 44 |
| Beyond the basics | 44 |
| Administrator's guide | 45 |
| Introduction | 45 |
| Terminology | 45 |
| Node types | 46 |
| Installation | 47 |
| System requirements | 47 |
| Installing Drupal | 48 |
| Easy way to install drupal with the cPanel control panel | 51 |
| General instructions | 52 |
| Special cases | 53 |
| Installing Drupal behind an Actiontec GT701-WG router | 53 |
| Installing Drupal in a subdirectory | 54 |
| More than one Drupal site on one machine | 55 |
| General rules for multiple Drupal deployments | 55 |
| Multiple directories | 55 |
| Multiple domains or vhosts | 56 |
| Multiple domains or vhosts using different databases | 56 |
| Multiple domains using the same database | 56 |
| Moving your Drupal installation to a new directory | 57 |
| Linux specific guidelines | 57 |
| Installing PHP, MySQL and Apache under Linux | 58 |
| Mac OS X-specific guidelines | 58 |
| Installing Drupal on Mac OS X 10.4 Tiger | 58 |
| Installing and Configuring MySQL | 59 |
| Creating the Drupal Database and Database User | 60 |
| Retrieving and Configuring Drupal | 61 |
| Windows-specific guidelines | 64 |
| How to install Drupal for newbies using FTP and phpMyAdmin | 64 |
| Change "/tmp" on your drupal site. | 64 |
| Get Drupal ready | 65 |
| Upload the database | 65 |
| Installing Apache (with PHP) on Windows | 66 |
| Installing MySQL on Windows | 66 |
| Installing PHP4 on Windows | 67 |
| Installing PostgreSQL on Windows | 67 |
| Untar | 68 |
| Using Clean URLs with IIS | 68 |
| Windows XP IIS development test system guidelines | 69 |
| Installing Drupal on Windows | 72 |
| Installing Drupal on Windows Ext | 73 |
| MS SQL Server Guidelines | 83 |
| PostgreSQL specific guidelines | 83 |
| Installing new modules | 84 |
| Installing new themes | 85 |

| | |
|---|-----|
| Tuning your server for optimal Drupal performance | 86 |
| Tuning PHP | 86 |
| PHP caches | 87 |
| Turck MMCache | 87 |
| Configuration | 88 |
| Settings | 89 |
| Anonymous user | 89 |
| Default front page | 89 |
| Examples | 90 |
| Clean URLs | 91 |
| Error pages | 92 |
| Error reporting | 92 |
| Cache support | 92 |
| File system settings | 93 |
| Download method | 93 |
| Path settings | 93 |
| Date and time settings | 94 |
| Customizing the interface | 94 |
| Customizing user login | 95 |
| Congestion control: tuning the auto-throttle | 95 |
| Adding syndicated content (newsfeeds, RSS) to your site | 99 |
| Database table prefix (and sharing tables across instances) | 101 |
| Advanced usage | 101 |
| Super advanced usage | 102 |
| Helping search engines index your site | 102 |
| Controlling what gets indexed -- the robots.txt file | 103 |
| PHP page snippets: putting dynamic content into your main page | 104 |
| How to insert and use the PHP Snippets in your pages | 105 |
| More sophisticated layout/styling of node pages | 105 |
| How to use the PHP Snippets with the front_page.module | 106 |
| USING PHP SNIPPETS WITH THE FRONT_PAGE MODULE | 106 |
| A guide to submitting your own PHP snippets | 106 |
| Countdown (x) days to a specific date and display a dynamic message | 107 |
| Create a list of node titles of a specific type, within certain dates/times | 107 |
| display (x) random thumbnails in a page | 108 |
| Display a list of (x) most recent weblog entries | 109 |
| Display a list of (x) node titles of a specific type | 109 |
| Display a list of (x) upcoming events in a scrolling box without javascript | 110 |
| Display a list of a certain content type, with teasers | 111 |
| Display a list of category titles with links to the full term | 111 |
| Display a list of node titles from a specific category | 112 |
| Display a list of the next (x) upcoming events | 113 |
| Display different page content to anonymous and authenticated users | 113 |
| Display the (x) most recent nodes in full from a specific category | 114 |
| Display the (x) most recent weblog entries from a specific user | 114 |
| Display the (x) most recent weblog entries with teasers & info. | 115 |
| Display/hide content from a specific IP address within a page | 116 |

| | |
|--|-----|
| Insert a quicklist of recent forum topic titles and links | 117 |
| Insert an image before the promoted nodes on the frontpage | 117 |
| Insert the most recent poll | 118 |
| maintenance redirect: automatically redirecting everyone but you away from the drupal site | 118 |
| Printing PHP variables from GET or POST forms | 119 |
| Using more than one php snippet in the same node (or front_page) | 120 |
| why? | 120 |
| Redirecting your host.example.com to a specific page | 121 |
| The tolerant base URL | 122 |
| Blocks | 123 |
| Block configuration | 123 |
| Restricting blocks to certain pages | 124 |
| Example: preventing a block from appearing | 124 |
| Custom blocks | 127 |
| Examples | 127 |
| A "book-like" navigation block | 128 |
| Latest stories block | 129 |
| Show block to certain users only | 129 |
| Show last 5 nodes by users on their profile pages | 131 |
| Submission queue block | 131 |
| Debugging the path | 132 |
| Custom login | 133 |
| All published content in a list | 133 |
| Blog categories | 134 |
| Blogger style 'About Me' block | 134 |
| Categories block as in 4.5 | 135 |
| Blogcentric random image | 136 |
| Hits by month | 136 |
| Comment approval count block | 137 |
| Counter (x days before / past...) | 138 |
| Paypal blocks | 138 |
| Show highest contributers to a site | 139 |
| Top users by comment number | 139 |
| Uptime and load on Unix systems | 140 |
| Pulldown top level category links | 140 |
| Drupal modules and features | 142 |
| Aggregator: syndicating content | 142 |
| Old page | 142 |
| What do I need to subscribe to a feed? | 143 |
| Configuring news feeds | 143 |
| Creating categories in the aggregator | 144 |
| Tagging individual items in the aggregator | 144 |
| Using the news aggregator | 144 |
| RSS feed blocks | 145 |
| Archive: view content by date | 145 |
| Block: controlling content in the sidebars | 145 |

| | |
|---|-----|
| Module blocks | 146 |
| Administrator defined blocks | 146 |
| Blog: a blog for every user | 146 |
| HOWTO: Configure user blogs | 147 |
| What is a blog or weblog? | 147 |
| Making user blogs more accessible | 148 |
| Additional features | 148 |
| BlogApi: post from blog tools | 149 |
| Book: collaborative document publishing | 149 |
| Maintaining a FAQ using a collaborative book | 150 |
| Comment: allow comments on content | 151 |
| Old page | 151 |
| User control of comment display | 152 |
| Additional comment configurations | 152 |
| Notification of new comments | 152 |
| Comment moderation | 153 |
| Moderation votes | 153 |
| Moderator vote/values matrix | 154 |
| Creating comment thresholds | 154 |
| Initial comment scores | 154 |
| Contact: allow other users to contact you | 154 |
| Drupal: Drupal sites directory server | 155 |
| Old page | 155 |
| Filter: Input formats for user content | 156 |
| Forum: create threaded discussions | 156 |
| HOWTO: Create a forum | 156 |
| HOWTO: Create forum containers | 157 |
| Help: context sensitive information | 157 |
| Legacy: remapping of old-style URLs | 158 |
| Locale: multi-language support | 158 |
| HOWTO: Creating a customized language set to replace Drupal terminology | 159 |
| Menu: customize site navigation | 159 |
| Node: the content | 160 |
| Page: post static pages | 160 |
| Path: readable URL's | 161 |
| Mass URL aliasing | 162 |
| Ping: notify services of changes | 162 |
| Poll: community voting | 163 |
| Profile: extending user account information | 163 |
| Enabling user pictures (avatars) | 163 |
| HOWTO: Create new profile fields | 164 |
| Search: an internal site search system | 164 |
| Statistics: tracking referrers, page hits, etc. | 164 |
| Old page | 165 |
| Introduction | 165 |
| Configuring the statistics module | 166 |
| Popular content block | 166 |

| | |
|---|-----|
| Story: post static pages | 166 |
| System: cron and caching | 167 |
| Configuring cron jobs | 167 |
| Taxonomy: categories and classification schemes | 168 |
| Vocabularies and terms | 169 |
| Creating a vocabulary | 170 |
| Creating terms | 170 |
| Advanced taxonomies: using hierarchies | 171 |
| Using vocabularies for navigation | 171 |
| More about Taxonomy | 173 |
| Creating a Block with links belonging to certain taxonomy terms | 173 |
| Throttle: congestion control | 174 |
| Tracker: viewing new and updated content | 174 |
| Upload: collaborate with files | 174 |
| Watchdog: monitor your site | 175 |
| User: access and management settings | 175 |
| Managing access control with permissions and user roles | 176 |
| Assigning permissions and users to roles | 177 |
| Adjusting permissions after adding modules | 177 |
| User authentication | 178 |
| User preferences and profiles | 178 |
| Using distributed authentication | 178 |
| Distributed authentication | 178 |
| Drupal | 179 |
| Contributed modules | 179 |
| Buddylist: list your social network | 179 |
| Event: set times for content | 180 |
| Flexinode: new content types | 180 |
| FOAF: friends of a friend | 181 |
| Fontsize: make text larger | 181 |
| Forms: forms for modules | 181 |
| HTMLarea | 182 |
| Image: galleries of images | 182 |
| Interwiki: wiki syntax for linking | 182 |
| Listhandler: connect mailing lists to forums | 183 |
| Location: associate geographic location | 183 |
| Mailalias: alias e-mails to user | 184 |
| Mailhandler: content via mail | 184 |
| Massmailer: manage mailing lists | 184 |
| Node import: get CSV content | 185 |
| Raw data import type | 185 |
| Node privacy by role: node view and edit permissions | 186 |
| Notify: email notification of new site content | 186 |
| Privatemsg: an internal messaging system | 187 |
| Queue: moderation, collaborative rating | 187 |
| Moderation queue | 187 |
| Comment rating | 188 |

| | |
|---|-----|
| RSVP: invite people | 188 |
| Survey: community questions | 188 |
| Taxonomy menu: navigation for terms | 189 |
| Textile: simple text syntax | 189 |
| Theme editor: store, configure, create | 190 |
| TinyMCE: a WYSIWYG editor | 190 |
| Trackback: post remotely | 191 |
| URL filter: turn URLs and e-mail addresses into live links | 192 |
| Volunteer: organize people to help | 192 |
| Upgrading from previous versions | 192 |
| Upgrading from Drupal 2.00 to 3.00 | 192 |
| Upgrading from Drupal 3.00 to 4.00 and later versions | 193 |
| Migrating from other weblog software | 193 |
| Migrating from ezPublish | 194 |
| Move ezp database content to drupal database | 194 |
| Parse ezxml (in perl, with LWP::UserAgent) | 197 |
| Get ezpublish user real names for drupal profile.module | 198 |
| Migrating from Geeklog | 198 |
| Migrating from LiveJournal | 199 |
| Import your LJ through an IFRAME held in a book page or similar | 200 |
| Using provided Import Module | 200 |
| Use the Livejournal Module to import the raw data into Drupal | 200 |
| Migrating from Movable Type | 201 |
| Extract Movable Type content as xml | 202 |
| Moving your MT styles and templates | 202 |
| Template for MT entry and comment export and Drupal import | 204 |
| Parse xml into sql insert statements | 205 |
| Insert content into Drupal nodes | 208 |
| Setting terms for inserted nodes | 208 |
| Migrating from PHPNuke | 209 |
| Migrating themes | 209 |
| Migrating users | 209 |
| Migrating from PostNuke | 209 |
| Configuring mod_rewrite in .htaccess for PN legacy URLs in | 209 |
| Search engine-friendly migration | 210 |
| Backups | 211 |
| Best practices guidelines | 212 |
| Accounts and roles | 212 |
| Backing Up Your Drupal Site | 212 |
| File/directory management | 213 |
| Test Sites | 213 |
| Troubleshooting FAQ | 213 |
| Installation/configuration | 214 |
| "Headers already sent" error | 214 |
| "LOCK TABLES sequences WRITE" error | 214 |
| "Method POST is not allowed for the URL /index.htm" error | 215 |
| .htaccess page forbidden | 215 |

| | |
|---|-----|
| Block referrer spam | 215 |
| E-Mail from Drupal is bouncing or not being sent | 216 |
| How can I administrate my navigation on my drupal site? | 217 |
| How can I install modules? | 218 |
| How do I unset the clean urls? | 219 |
| No content on main page for non-admin users | 219 |
| Permission denied in includes/file.inc | 219 |
| PHP safe mode issue | 220 |
| What is the minimum version of PHP? | 221 |
| Nodes | 221 |
| Can't create static php page | 221 |
| PHP content won't parse | 221 |
| Schedule and expire Nodes | 221 |
| Search | 221 |
| Search index db empty/incomplete | 221 |
| Search multibytes language | 222 |
| Polls | 222 |
| Are polls supported in Drupal? | 222 |
| Can a user vote more than once in a poll? | 222 |
| Miscellaneous | 222 |
| Download offline copy of drupaldocs.org - stuck without net access | 222 |
| How can I change Drupal's character encoding? (UTF-8 and Unicode) | 222 |
| How do I report a bug in a contributed module? | 223 |
| How to install a Patch? | 223 |
| Making a custom script work (independently) along with a Drupal setup | 223 |
| Move existing site to new server | 224 |
| Moving your site to another url | 224 |
| My URL is wrong in the list of Drupal sites | 225 |
| Truncated fields/unable to login/PHP 4.2.3 bug | 225 |
| Contributor's guide | 226 |
| Contributing to Drupal | 226 |
| Types of Contributions | 226 |
| Task list | 227 |
| Bug reports | 227 |
| How to report bugs effectively | 228 |
| Feature suggestions | 234 |
| Patches | 234 |
| Diff and patch | 234 |
| Diff on Windows | 236 |
| Patch on Windows | 237 |
| Creating and submitting patches | 238 |
| Rules of reviewing patches | 239 |
| The revision process | 240 |
| Criteria for evaluating proposed changes | 241 |
| Maintaining a project on drupal.org | 241 |
| Downloads and packaging | 242 |
| Managing releases | 242 |

| | |
|--|-----|
| Orphaned projects | 242 |
| Tips for contributing to the core | 243 |
| Mailing lists | 243 |
| Newsletter | 243 |
| Drupal-support | 243 |
| Drupal-devel | 244 |
| Drupal-docs | 244 |
| Drupal-cvs | 244 |
| Infrastructure | 244 |
| Mailing of project issues | 244 |
| Coding standards | 245 |
| Drupal Coding Standards | 245 |
| Indenting | 245 |
| Control Structures | 245 |
| Function Calls | 246 |
| Function Declarations | 246 |
| Comments | 246 |
| Including Code | 247 |
| PHP Code Tags | 247 |
| Header Comment Blocks | 247 |
| Using CVS | 247 |
| Example URLs | 248 |
| Naming Conventions | 248 |
| Functions and Methods | 248 |
| Constants | 248 |
| Global Variables | 248 |
| Filenames | 248 |
| Doxygen formatting conventions | 248 |
| Comments | 251 |
| Indenting | 251 |
| PHP Code tags | 251 |
| SQL naming conventions | 251 |
| Functions | 252 |
| Constants | 252 |
| Control structures | 252 |
| Header comment blocks | 253 |
| CVS | 253 |
| CVS concepts | 254 |
| Using CVS with branches and tags | 255 |
| Windows | 256 |
| Available Branches | 256 |
| Apply for contributions CVS access | 257 |
| CVS GUIs and clients | 257 |
| Cross-platform CVS clients | 257 |
| Eclipse CVS plug-in | 257 |
| CVS front ends for Windows | 257 |
| TortoiseCVS | 257 |

| | |
|---|-----|
| WinCVS | 258 |
| CVS on Mac OS X | 259 |
| CVL: point and click CVS | 259 |
| Setting up/step by step CVS | 259 |
| Basic CVS with CVL | 261 |
| Preparing a project | 261 |
| Committing a project | 262 |
| Drupal CVS repositories | 263 |
| Main repository | 263 |
| Contributions repository | 263 |
| Promoting a project to be an official release | 264 |
| Adding a file to the CVS repository | 264 |
| Tracking Drupal source with CVS | 265 |
| Example | 265 |
| Updating the vendor branch | 266 |
| Summary | 268 |
| Additional resources | 268 |
| Sandbox maintenance rules | 268 |
| Additional references | 269 |
| Drupal's APIs | 269 |
| Drupal.org site maintainers | 269 |
| Site maintainer's guide | 270 |
| Unpublishing vs deleting of content | 271 |
| Blocking vs deleting of users | 271 |
| Suggested Workflow | 271 |
| Badly formatted posts | 271 |
| Drupal test suite | 272 |
| FAQ | 272 |
| PHP Debugger | 272 |
| Module developer's guide | 273 |
| Introduction to Drupal modules | 273 |
| Drupal's menu building mechanism | 273 |
| Drupal's node building mechanism | 276 |
| How Drupal handles access | 280 |
| Drupal's page serving mechanism | 281 |
| Creating modules - a tutorial | 288 |
| Getting started | 288 |
| Letting Drupal know about the new function | 289 |
| Telling Drupal about your module | 289 |
| Telling Drupal who can use your module | 290 |
| Announce we have block content | 291 |
| Generate content for a block | 292 |
| Installing, enabling and testing the module | 295 |
| Create a module configuration (settings) page | 296 |
| Adding menu links and creating page content | 298 |
| Adding a 'more' link and showing all entries | 299 |
| Conclusion | 300 |

| | |
|---|-----|
| Updating your modules | 300 |
| Converting 3.0 modules to 4.0 | 300 |
| Converting 4.0 modules to 4.1 | 301 |
| Required changes | 301 |
| Optional changes | 302 |
| Converting 4.1 modules to 4.2 | 302 |
| Converting 4.2 modules to 4.3 | 304 |
| Creating modules for version 4.3.1 | 305 |
| Getting Started | 305 |
| Telling Drupal about your module | 306 |
| Telling Drupal who can use your module | 307 |
| Announce we have block content | 308 |
| Generate content for a block | 308 |
| Installing, enabling and testing the module | 312 |
| Create a module configuration (settings) page | 313 |
| Adding menu links and creating page content | 315 |
| Letting Drupal know about the new function | 316 |
| Adding a more link and showing all entries | 317 |
| Conclusion | 318 |
| How to build up a _help hook | 318 |
| How to convert a _system hook | 319 |
| How to convert an _auth_help hook | 321 |
| Converting 4.3 modules to 4.4 | 322 |
| Menu system | 322 |
| Theme system | 324 |
| Node system | 324 |
| Filter system | 325 |
| Hook changes | 326 |
| Emitting links | 327 |
| Status and error messages | 327 |
| Converting 4.4 modules to 4.5 | 328 |
| Menu system | 328 |
| Path changes | 329 |
| Node changes | 329 |
| Filtering changes | 331 |
| Check_output() changes | 331 |
| Filter hook | 331 |
| Filter tips | 333 |
| Other changes | 333 |
| Converting 4.5 modules to 4.6 | 333 |
| Block system | 333 |
| Search system | 334 |
| Module paths | 334 |
| Database backend | 335 |
| Theme system | 335 |
| Watchdog messages | 335 |
| Node markers | 335 |

| | |
|---|-----|
| Control over destination page after form processing | 335 |
| Confirmation messages | 336 |
| Inter module calls | 336 |
| Node queries | 336 |
| Text output | 337 |
| Converting 4.6 modules to HEAD | 338 |
| Taxonomy API change | 338 |
| Table API change | 338 |
| Join forces | 339 |
| Reference | 339 |
| 'Status' field values for nodes and comments | 339 |
| Values of 'comment' field in node table | 340 |
| Module how-to's | 340 |
| How to write a node module | 340 |
| How to write database independent code | 340 |
| How to write efficient database JOINs | 341 |
| How to connect to multiple databases within Drupal | 342 |
| How to write themable modules | 343 |
| Theme developer's guide | 345 |
| Theming overview | 345 |
| Creating custom themes | 346 |
| PHPTemplate theme engine | 346 |
| Installing PHPTemplate | 347 |
| Creating a new PHPTemplate | 347 |
| Block.tpl.php | 348 |
| Available variables | 348 |
| Default template | 348 |
| Box.tpl.php | 348 |
| Available variables | 349 |
| Default template | 349 |
| Comment.tpl.php | 349 |
| Available variables | 349 |
| Default template | 349 |
| Node.tpl.php | 350 |
| Available variables | 350 |
| Default template | 350 |
| Theme distinct node types differently | 351 |
| Page.tpl.php | 351 |
| Available variables | 351 |
| Default template | 352 |
| Alternative templates for different node types | 354 |
| Example - Theming flexinode | 354 |
| The Quick Version | 355 |
| Create template.php | 355 |
| Create flexinode_timestamp.tpl.php | 355 |
| The Long Version | 356 |
| 1. find the theme function for the flexinode field | 356 |

| | |
|---|-----|
| 2. Create template.php and add override function | 356 |
| 3. Create flexinode_timestamp.tpl.php to do formatting | 357 |
| Example Files | 357 |
| template.php | 358 |
| flexinode_timestamp.tpl.php | 358 |
| Making additional variables available to your templates | 359 |
| Overriding other theme functions | 360 |
| Example - Overriding the user profile pages using PHPTemplate | 361 |
| Before | 361 |
| After | 362 |
| Not including drupal.css | 362 |
| Protecting content from anonymous users when using overrides | 362 |
| Using PHPTemplate Overrides with protected content | 362 |
| Example | 362 |
| Solution | 363 |
| Themeing front page and others | 363 |
| XTemplate to PHPTemplate conversion | 364 |
| XTemplate theme engine | 366 |
| Creating a new XTemplate | 366 |
| Template basics | 367 |
| Section Tags | 367 |
| Item Tags | 367 |
| Header section | 368 |
| The Section | 368 |
| Prolog | 368 |
| DOCTYPE | 369 |
| {head_title} | 369 |
| {head} | 369 |
| {styles} | 369 |
| {onload_attributes} | 370 |
| {logo} | 370 |
| {site_name} | 370 |
| {site_slogan} | 370 |
| {secondary_links} {primary_links} | 370 |
| Search Box | 371 |
| {search_url} | 371 |
| {search_description} | 371 |
| {search_button_text} | 371 |
| Mission | 371 |
| {mission} | 371 |
| Title | 371 |
| {title} | 372 |
| Tabs | 372 |
| {tabs} | 372 |
| {breadcrumb} | 372 |
| Help | 372 |
| {help} | 372 |

| | |
|-------------------------------|-----|
| Message | 372 |
| {message} | 372 |
| Node section | 373 |
| The Node Section | 373 |
| {sticky} | 373 |
| Picture | 373 |
| {picture} | 373 |
| Title | 373 |
| {link} | 374 |
| {title} | 374 |
| {submitted} | 374 |
| Taxonomy | 374 |
| {taxonomy} | 374 |
| {content} | 374 |
| Links | 374 |
| {links} | 375 |
| Comment | 375 |
| The Comment Section | 375 |
| Avatar | 375 |
| {avatar} | 375 |
| Title | 375 |
| {link} | 376 |
| {title} | 376 |
| Submitted | 376 |
| {submitted} | 376 |
| New | 376 |
| {new} | 376 |
| Content | 376 |
| {content} | 376 |
| Links | 376 |
| {links} | 377 |
| Blocks | 377 |
| The Section | 377 |
| {blocks} | 377 |
| Block | 377 |
| {module} | 377 |
| {delta} | 377 |
| {title} | 377 |
| {content} | 378 |
| Footer | 378 |
| The Footer Section | 378 |
| Message | 378 |
| {footer_message} | 378 |
| {footer} | 378 |
| Editing with Golve | 378 |
| Set Up | 378 |
| Editing | 379 |

| | |
|---|-----|
| Plain PHP themes | 379 |
| Theme coding conventions | 381 |
| Updating your themes | 382 |
| Converting 3.0 themes to 4.0 | 382 |
| Required changes | 382 |
| Changes in class definition | 382 |
| Changes in function header() | 382 |
| Changes in function node() | 383 |
| Changes in function footer() | 384 |
| Optional changes | 384 |
| New function: system() | 384 |
| Converting 4.0 themes to 4.1 | 384 |
| Required changes | 384 |
| Optional changes | 385 |
| theme_head | 385 |
| Converting 4.1 themes to 4.2 | 385 |
| Required changes | 385 |
| Add a theme_onload_attribute() to a <body> tag: | 385 |
| Optional changes | 385 |
| Take advantage of settings() hook | 385 |
| Direct you site logo to index.php | 385 |
| Converting 4.2 themes to 4.3 | 386 |
| Converting 4.3 themes to 4.4 | 386 |
| Converting 4.4 themes to 4.5 | 389 |
| Directory structure | 389 |
| Tabs (a.k.a. Local Tasks) | 389 |
| Status Messages | 390 |
| Static vs. Sticky | 390 |
| Avatar vs. User Picture | 391 |
| Theme Screenshots | 392 |
| Centralized Theme Configuration | 392 |
| Styles | 394 |
| _help hook | 394 |
| Converting 4.5 themes to 4.6 | 394 |
| Search form | 394 |
| Node links | 395 |
| Pages | 395 |
| Node and comment markers | 395 |
| Pager and menu item themeing | 395 |
| Text validation changes | 395 |
| Converting 4.6 themes to HEAD | 396 |
| Table row coloring | 396 |
| Theme screenshot guidelines | 396 |
| Theme how-to's | 398 |
| Tips for designing themes in Dreamweaver, GoLive etc. | 398 |
| Dreamweaver | 398 |
| In Extensions.txt | 398 |

| | |
|---|-----|
| In MMDocumentTypes.xml | 398 |
| Adding your theme to Drupal.org | 399 |
| Theme snippets repository | 399 |
| Custom login | 399 |
| Customize display of submission information based on node type | 400 |
| How to display mission on every page? | 400 |
| Make images square | 400 |
| Overriding drupal.css; two approaches | 401 |
| Documentation writer's guide | 402 |
| Getting involved in documentation projects | 402 |
| Requests for new documentation and reporting documentation problems | 402 |
| Creating or updating pages in the Drupal handbooks | 402 |
| How to add a page to the Handbook | 403 |
| Authoring guidelines | 403 |
| Adding screenshots | 405 |
| Avoid all embedded headings | 406 |
| HOWTO: Writing admin/help documentation | 406 |
| Resources for getting started | 407 |
| Guidelines for structure, content, and links | 407 |
| Modifying the handbook modules pages for inclusion in Drupal core | 408 |
| Spelling and capitalization | 408 |
| Responsibilities of documentation project teams | 409 |
| Documentation project proposals and specs | 409 |
| Translator's guide | 411 |
| Translation templates | 411 |
| Programs to use for translation | 412 |
| Issues using poEdit | 412 |
| Plurals Solution #1 | 412 |
| Plurals Solution #2 | 413 |
| Plurals Solution #3 | 413 |
| Setting up XEmacs with po-mode on Windows | 416 |
| Translated Drupal information | 417 |
| African | 417 |
| Vir diegene wat betrokke wil raak by die vertaling: | 417 |
| Om 'n fout met die vertaling te rapporteer: | 417 |
| Vir enige ander verwante redes waaroor jy wil kontak: | 417 |
| Russian | 417 |
| Spanish | 418 |
| Translation guidelines | 418 |
| Translation of contributed modules | 418 |
| Distributing the translation effort | 419 |
| Status of the translations | 419 |
| Status overview | 419 |
| Checking your translation status | 420 |
| Translations for contributed modules | 421 |
| Make a single file from the loose .po files from CVS | 421 |
| Recycling old translations | 421 |

| | |
|---|-----|
| Troubleshooting | 422 |
| Weird characters or question marks | 422 |
| Marketing resources | 423 |
| Banners and buttons | 423 |
| Official buttons | 423 |
| Others | 423 |
| Links | 424 |
| Logos for special events | 424 |
| Booklet | 424 |
| Druplicon | 425 |
| Bitmap versions | 425 |
| Vector formats | 426 |
| Other formats | 427 |
| Logo colors | 427 |
| Presentations | 427 |
| Reviews | 428 |
| Posters | 428 |
| Drupal t-shirts and other merchandise | 429 |
| Basic Druplicon | 429 |
| Designed by Michael Angeles | 429 |
| About the handbook | 431 |
| Acknowledgements | 431 |
| Commenting on the handbook pages | 431 |
| Copyright and licensing | 432 |
| Drupal documentation project teams | 432 |
| Press and Marketing | 433 |
| Technology initiatives | 435 |
| Recent updates | 435 |
| Book contributors | 438 |

Drupal handbook

This document offers a complete reference for those interested in Drupal, both novice and experienced Drupal administrators, Drupal users and Drupal developers.

The Drupal handbook is © copyright 2000-2005 by the individual contributors and can be used in accordance with the Creative Commons License, Attribution-ShareAlike 2.0.

About Drupal

Drupal is software that allows an individual or a community of users to easily publish, manage and organize a great variety of content on a website. Tens of thousands of people and organizations have used Drupal to set up scores of different kinds of web sites, including

- community web portals and discussion sites
- corporate web sites/intranet portals
- personal web sites
- aficionado sites
- e-commerce applications
- resource directories

Drupal includes features to enable

- content management systems
- blogs
- collaborative authoring environments
- forums
- newsletters
- picture galleries
- file uploads and download

and much more.

Drupal is open source software licensed under the GPL, and is maintained and developed by a community of thousands of users and developers. Drupal is free to download and use. If you like what Drupal can do for you, please work with us to expand and refine Drupal to suit your needs.

Is Drupal right for you?

If you are new to Drupal, you will probably have many questions about its capabilities and suitability for your applications. The information in this section is intended to help you decide whether Drupal is appropriate for your needs.

The Case studies section examines typical uses for Drupal, and presents some examples of sites using Drupal for each of the types of web-site discussed. This section includes a listing of hundreds of known Drupal sites.

In the Feature overview we survey some of the most important and commonly deployed features of Drupal.

A discussion of the merits of using Drupal over writing a custom Web-application framework to support your project is presented in Rolling your own system vs. using Drupal

Gallery

Take a moment to visit some exceptional Drupal websites that exhibit the versatility of this tool. These sites display how Drupal sites can be built to meet many different functions and needs while still being graphically appealing and dynamic.

acko.net



Cam Oca&A±



Can't Eat It



Copenhagen



Drupal Hungary



Echo Ditto



Eric Scouten Photography



Footnotes



Fulbright - EspaÃ±a



Hip Foto



Kernel Trap



Our Media



Pop Madrid



Snowboard Magazine



Spread Firefox



Terminus 1525



URLGreyHot



Case studies

Drupal meets the needs of different types of web sites:

Community Portal Sites If you want a news web site where the stories are provided by the audience, Drupal suits your needs well. Incoming stories are automatically voted upon by the audience and the best stories bubble up to the home page. Bad stories and comments are automatically hidden after enough negative votes. *Examples:* Debian Planet | Kerneltrap

Personal Web Sites Drupal is great for the user who just wants a personal web site where she can keep a blog, publish some photos, and maybe keep an organized collection of links. *Examples:* urlgreyhot | Langemarks Cafe

Aficionado Sites Drupal flourishes when it powers a portal web site where one person shares their expertise and enthusiasm for a topic. *Examples:* ia/ | Dirtbike

Intranet/Corporate Web Sites Companies maintain their internal and external web sites in Drupal. Drupal works well for these uses because of its flexible permissions system, and its easy web based publishing. No longer do you have to wait for a webmaster to get the word out about your latest project. *Examples:* Sudden Thoughts | Tipic

Resource Directories If you want a central directory for a given topic, Drupal suits your needs well. Users can register and suggest new resources while editors can screen their submissions. *Example:* Entomology Index

International Sites When you begin using Drupal, you join a large international community of users and developers. Thanks to the localization features within Drupal, there are many Drupal sites implemented in a wide range of languages. *Examples:* PuntBarra | cialog

Sites that use Drupal

The following list of sites is compiled dynamically. All Drupal installations may optionally ping a Drupal directory server via XML-RPC. This server collects all active installations and displays them here. Currently, there are Drupal sites in our database but this is only a fraction of all Drupal sites. More information is available on this page in the Drupal handbook.

Success stories

This part is dedicated to real-life examples of how drupal can help to solve your business problems. Please share the success of your drupal implementation.

Contaire.com: a corporate website based on Drupal

Drupal is well suited for community plumbing, alright. But what if you want to apply its power, elegance, and simplicity to your corporate website? In this article, we explain our approach to creating a corporate website with Drupal, show you how to create your templates and arrange your content. So why would you want to enter into such a formidable endeavor?

Note: the original article is on contaire.com's site, and contains additional screenshots and illustrations.

- Both the layout and the underlying HTML of our old website needed a face lift.
- We specialize in sophisticated content management solutions, yet our website consisted of static HTML pages with an absolute minimum of PHP code to avoid the worst code duplication. We knew we could do better.
- At times we were slow to post updates of our site. The process of doing so should be more straight forward.

Our requirements

Our requirements were quickly set:

- The site layout should remain largely as is, with two thirds for the main content area and a column of news headlines on the right.
- There is a flat list of sections with articles, some of which arrange the teasers in two, others in a single column.
- The sections can be accessed by a dynamical list of tabs at the top of the page.
- Sections should have simple URLs, with some other articles having intuitive URLs as well.
- The page structure is such that the front page features article or section teasers.
- Initially, there will be no community features like comments.
- Content should be editable through its front end view.
- The site should validate as XHTML, and should further follow the guidelines for barrier

free web sites.

- Text formatting should be using Textile markup.
- Technically, there should be as little as possible software development on top of stock Drupal. However, we wanted to develop templates using the PHPTAL engine and knew we would have to transform one of the standard Drupal templates to this notation first.

The ingredients

We started development with the following ingredients:

- "Drupal 4.5.1":<http://drupal.org/files/projects/drupal-4.5.1.tar.gz>
- The contributed modules collimator.module
(<http://drupal.org/files/projects/collimator-4.5.0.tar.gz>), image.module
(<http://drupal.org/files/projects/image-4.5.0.tar.gz>), image_filter.module
(http://drupal.org/files/projects/image_filter-4.5.0.tar.gz), and textile.module
(<http://drupal.org/files/projects/textile-4.5.0.tar.gz>)
- Our own contributed theme engine phptal.engine
(<http://drupal.org/files/projects/phptal-cvs.tar.gz>)
- The stock Marvin theme

A dynamic horizontal tab menu

The most prominent feature here is the horizontal navigation tabs. This has become a popular arrangement recently, very often enhanced with drop-down menus. In our case, there are no drop-down menus. The underlying implementation, however, should be easily extended to host these as well.

Three standard features of Drupal, a PHP theme function and a little CSS magic are used to implement the horizontal tab menu. The features are

- Taxonomies. We use a separate vocabulary "Sections" to organize content.
- The menu is not linked directly to this vocabulary as would the Drupal module taxonomy_menu.module do, but rather is created through a customized menu. This allow linking menu entries to taxonomy pages, individual nodes, or two column pages generated by the collimator.module.
- Finally, links in the menu are cleaned by assigning URL aliases to menu entries.

For example, the entry "partner" links to "partner":/partner which is an alias for "collimator/4":/collimator/4, i.e., the two column listing of teasers for topic 4 ("Lebendiges Netzwerk").

What remains is a function that renders the menu:

```
<?php
function _contaire_menu($pid = 1) {
  $menu = menu_get_menu();
  $entries = array();
  if (isset($menu['visible'][$pid]) &&
```

```

$menu['visible'][$pid]['children'])
{
  foreach ($menu['visible'][$pid]['children'] as $mid) {
    $style = (count($menu['visible'][$mid]['children'])) ?
menu_in_active_trail($mid)
      ? 'expanded' : 'collapsed'
      : 'leaf');
    $entry = array('style' => $style, 'link' => theme('menu_item',
$mid));
    $entry['kids'] = _contaire_menu($mid);
    $entries[] = $entry;
  }
}
return $entries;
}
function contaire_menu($pid = 1) {
  return _phptal_callback('_menu',
    array('pid' => $pid, 'entries' => _contaire_menu($pid)));
}
?>

```

In the PHPTAL theme engine we use, this function can be written into the @template.php@ file of our theme and be called from the file @page.tal@ as

```

<div id="header">
  ...
  <div tal:content="php:contaire_menu(26)" />
</div>

```

Here is the menu entry for our custom menu.

A new teaser.module

Except for the horizontal navigation, the front page looks like standard Drupal, but looking more closely, even here there are interesting details:

- Each teaser has a picture associated with it.
- The teaser pictures float automatically to the left and right.
- The "weiter" links are placed behind, not below the teaser texts.

There is one feature not visible on the site as presented to the public that are little edit buttons placed to the right of the headlines. These become necessary because we haven't linked our headlines to a detail view with tabbed local tasks, and

- the "weiter" link may link to an arbitrary URL.

We have created "a small Drupal module":<http://drupal.org/node/14920> to provide these features.

Two columns, but sorted, please

Porting our original content we needed a way to layout some pages in two columns. One example is our partners page (<http://contaire.com/partner>) where in addition to the two columns we have an introductory text at the top. We quickly settled on the contributed collimator.module but had to patch it to give us control over the way it sorts articles. The collimator offers the standard modes by-date and by-title to sort. We wanted to control sorting explicitly and abused a new node field _teaser_weight_ for this. The _abuse_ here is that actually this field should be a property of the table term_data but there is no easy way to add this without patching the taxonomy.module. We provided our changes as a patch (<http://drupal.org/node/15240>) to the collimator.module.

The single column text at the page top is simply the description of the page's taxonomy term, fed through Textile.

Conclusion

We just love our new page! Once we decided on the selection of modules and exactly for which features we would have to write some code the actual effort was well worth it. The phptal.engine has had its first live test and proved fun to work with.

All in all, Drupal again showed its greatness and that - with a little thought and planning - it can be used for many a corporate website.

Why Linux Journal converted to Drupal and how it went

We had been looking for a "Content Management System" for quite a while, and one of our employees discovered Drupal while researching CMS software on the web. Drupal appeared to be much more flexible than PHP Nuke, which we were using, and the more we looked at it the more impressed we became. At that time all of the features we thought we would need, except one, which we decided to write a module to provide, were on the table to be implemented in the next Drupal release.

We decided to convert Linux Gazette to Drupal in order to become familiar with Drupal under real life high usage conditions and to setup Doc Searls' IT Garage to experiment with Drupal's blogging and other interactive abilities. After we were satisfied with Drupal's ability to handle the traffic at Linux Gazette as well as its interactive abilities at Doc Searls' IT Garage and had set up and tested Drupal's flexibility by creating several sites for internal corporate use, we decided to create a Drupal site to replace the Linux Journal PHP Nuke site that we were using for Linux Journal. We started out with version 4.3 of Drupal but by the time we decided to convert Linux Journal to Drupal, version 4.4 was out so we started building our new site using that version. There were not any major problems, just the typical learning curve requiring new ways of looking at problems, nothing we could see that would prevent us from using Drupal for the new site. Most of our time was devoted to developing methods to convert the old articles and content of Linux Journal magazine to the new format required in the new site.

One thing that Drupal did not provide was a method that we could use to display static content in the main center section without the other content that Drupal normally puts there. We wanted to be able to link to static html and text files and have only that file be displayed in the center section. To do that Mitch Frazier created a module he called xstatic.module. The xstatic module allows you to define one or more base directories that can be used for storing HTML files, PHP files, text files, and image files. When the xstatic module is used in the URL, the argument to it is interpreted as a file path name. This file path name is searched for relative to all each of the predefined xstatic directories. The file type extensions are appended automatically. If the file is found its contents are displayed in the main (center) section of the Drupal page. If the file is a php file it is first evaluated and the result is displayed. This allowed us to create anything we wanted in the center, without having to create a node, while maintaining a consistent Drupal "look" with the site's header, sidebars, and footer intact. The xstatic module gave us a great way to separate all information that is not editorial content from the marketing and business oriented pages as well as providing us with a simple way to quickly integrate existing HTML files into the site.

After converting 10 years of articles and getting the "look and feel" we wanted, we decided to do the roll over from PHP Nuke to Drupal at 8am on Nov. 1st. Unfortunately on the evening of Oct. 30, while doing the final move of the articles to the new site, we discovered that Anonymous users could not leave their name or email address when making comments. This was a feature we "had" to have and was only available on version 4.5. On earlier versions of Drupal you had to have an account before your name would appear with your comment. This was a "show stopper" and even though we had less than 48 hours to do it in, we decided to install a new 4.5 site and bring over the blocks, theme and other changes we created for the earlier version. Mitch Frazier had it working in 24 hours and we spent the rest of the time before the roll out testing and doing minor cleanup.

The new site has been well received by Linux Journal subscribers and www.linuxjournal.com readers. Lots of helpful suggestions have been made and new features implemented because of them. Because of this warm reception, when we decided to create a publication called TUX, which is primarily for the new Linux user, we decided to use Drupal for its web site. Since we were short of time, we simply cloned the Linux Journal site. We then made cosmetic changes and cleaned up the database. This allowed us to have a working site while we worked on a completely new layout and design. The new TUX layout and design has been finished and is now in place. Steven Wittens, one of the core Drupal developers helped us with the look and feel of the new TUX site which is based on the phptemplate theme.

On the Linux Journal site, Drupal version 4.5 is handling 400,000 hits per day, and MySQL is handling the storage and the searches for 5,000 articles and over 14,000 comments. We are currently using these contributed modules; print, spam, subscriptions and themedev on both the Linux Journal and TUX sites. We are also considering using weblink, userpost and webforms as well. We are very pleased with the power and stability of Drupal and because of this are creating internal Drupal sites to be used for information dispersal and coordination between employees and departments. We are considering using a node level permission module that Matt Westgate is developing to control access to information in these internal sites. We are constantly amazed at how versatile and powerful Drupal is and at the new uses we find for it.

The Drupal Community has been a great help in answering questions and making suggestions that allowed us to create, design and convert our existing web site to Drupal as well as create new ones. To return the favor we are planning on releasing the xstatic module that Mitch Frazier created, some time after the first of the year. Mitch is also working on a few other new ideas and we will be releasing them after they are fully developed and researched.

Many thanks to the Drupal Community from the staff of Linux Journal, TUX, Doc Searls' IT Garage and Linux Gazette.

-- Keith Daniels
Web Coordinator
SSC Publications, Inc.
Publishers of:
Linux Journal
TUX
Doc Searls' IT Garage
Linux Gazette
A42
Groups of Linux Users Everywhere

Feature overview

Typical Drupal applications

By enabling and configuring individual modules, an administrator can design a unique site, one which can be used for a combination of knowledge management, web publishing and community interaction purposes. So that you can better understand the many possibilities, the following list of features have been organized by common web platform characteristics:

- **Content management.** Via a simple, browser-based interface, members can publish to a number of available content modules: stories, blogs, polls, images, forums, downloads, etc. Administrators can choose from multiple theme templates or create their own to give the site a singular look and feel. The flexible classification system allows hierarchical classifications, cross-indexing of posts and multiple category sets for most content types. Access to content is controlled through administrator-defined user permission roles. Site pages can display posts by module type or categorized content, with separate RSS feeds available for each display type. Users can also keyword search the entire site.
- **Weblog.** A single installation can be configured as an individual personal weblog site or multiple individual weblogs. Drupal supports the Blogger API, provides RSS feeds for each individual blog and can be set to ping weblog directories such as blo.gs and weblogs.com when new content is posted on the home page.
- **Discussion-based community.** A Drupal site can act as a Slashdot-like news site and/or make use of a traditional discussion forum. Comment boards, attached to most content types, make it simple for members to discuss new posts. Administrators can control whether content and comments are posted without approval, with administrator approval or through community moderation. With the built-in news aggregator, communities can

subscribe to and then discuss content from other sites.

- **Collaboration.** Used for managing the construction of Drupal, the project module is suitable for supporting other open source software projects. The wiki-like collaborative book module includes versioning control, making it simple for a group to create, revise and maintain documentation or any other type of text.

For a more comprehensive feature list, consult our feature overview. For live examples of possible site implementations, see the featured sites included with the Drupal case studies. Or visit some of the many sites that use Drupal.

Rolling your own system vs. using Drupal

Some of you might consider rolling your own system instead of using Drupal. As the Drupal community is very interested in having you join us instead, we will, for your consideration, present you with some advantages of using Drupal over rolling your own system. We encourage you to consider the following before setting off on your own.

The Drupal path is fun and educational

As you struggle to get your sites to fit your needs, Drupal will stimulate you in a more rewarding way than would a more low-level environment such as a plain programming language. As you go along you will experience and learn the inner workings of a system which has proven very powerful and which has become very popular. This will be useful for you in your later projects.

Dealing with insignificant details when writing a framework can be frustrating. With Drupal, the framework is in place and you can focus on the 'meat' of your project. With Drupal, you get the job done with less pain.

Drupal's user and developer base

Drupal is used by tens of thousands of sites on the web, and more than seven hundred people are visiting drupal.org as I write this. This gives you several advantages if you take the Drupal path:

- **Modules for a wide range of needs:** As you build your site, needs may arise which you didn't predict. As users of Drupal experience needs, a module is often written to fill it. In most cases these are contributed, so with the Drupal path you have the opportunity to download modules and easily plug these into a system which is able to handle them. Furthermore, you have the opportunity to hack the modules to fit your needs better as they are open source.
- **Easy access to help:** As mentioned, lots of people are active on drupal.org. Letting them know of your problems in the forum or searching there will often result in suggested solutions. When rolling your own system, you don't have the advantage of a large community "speaking the language" of your framework.
- **A thoroughly tested platform:** As the complexity of a system grows, it is likely to create errors or bugs. It will eventually be difficult, if not impossible for mere human to predict all

eventualities. You can not count on your site's visitors to report errors, because instead of reporting they may chose to leave.

Every installed Drupal site comes with a watchdog module. This module logs errors and report them to the administrator when she visits the administration section of her site. Furthermore it is important for all users of Drupal that their sites work as expected. Therefore users report errors to the maintainers of Drupal and her modules. To ease this process, Drupal's maintainers have created a system which keeps track of issues, showing them to interested persons able to correct them.

With Drupal, you are able to create a cool site

I've collected some example sites to illustrate this point. Notice that they are quite diverse.

- Our media
- Varal
- Tipic
- Urlgreyhot
- Political physics
- Bryght

More examples can be found in case studies and in this comprehensive list of drupal sites.

Drupal is flexible

A reason for rolling your own system might be to have it fit your needs exactly. However, Drupal is designed to fit a range of needs, one of her key features is flexibility. Chances are you're better off towards your goal with Drupal. In addition to the already mentioned modules and their ability to be adapted to fit your special needs, it is fairly easy to roll your own modules. This is typically done by making a file with some functions implementing certain hooks in addition to other functions. The following function implements the help hook which is called by the Drupal core and possibly some other Drupal modules.

```
<?php
function mymodule_help($section) {
  switch($section) {
    case 'admin/help#block':
      return 'My module will help you get laid';
      break;
  }
?>
```

This way modules and the Drupal core interact, and it has proved very powerful. By using hooks, modules can interact with and take advantage of the building blocks of Drupal such as her node, category, administration and user systems.

You can grow in the future with Drupal

Drupal will continue to be maintained in the future. This is important for sites adaption to the environment in which they live, the users they interact with and the software on which they run, as it is in constant flux. New needs will probably arise as your site grows. Perhaps one day a new protocol might be widely used, and your site need to support it. RSS support is an example of something many sites have needed to implement lately. When needs like these arise, someone probably need to dive into the code and adapt it to its changed environment. If they haven't been there for a while or are new to the code, things typically turn out to be complicated. The fact that people don't like to comment their code contribute to this. "Maintenance nightmare" is a phrase commonly used for these kind of things, as a search on google suggest.

However, if you take the Drupal path, chances are there will be a solution at drupal.org.

Background

History

In 2000, permanent Internet connections were at a premium for University of Antwerp students, so Dries Buytaert and Hans Snijder setup a wireless bridge between their student dorms to share Hans's ADSL modem connection among eight students. While this was an extremely luxourous situation at that time, something was missing. There was no means to discuss or share simple things.

This inspired Dries to work on a small news site with a built-in webboard, allowing the group of friends to leave each other notes about the status of the network, to announce where they were having dinner, or to share some noteworthy news items.

The software did not have a name until the day after Dries moved out after graduation. The group decided to put the internal website online so that they could stay in touch, continue to share interesting findings, and narrate snippets of their personal lives. While looking for an appropriate domain name, Dries settled for 'drop.org' after he made a typo to see if the name 'dorp.org' was still available. Dorp is the Dutch word for 'village', which was considered an appropriate name for the small community.

Once established on the Web, drop.org's audience changed as the members began talking about new web technologies such as moderation, syndication, rating, and distributed authentication. Drop.org slowly turned into a personal experimentation environment, driven by the discussions and flow of ideas. The discussions about these web technologies were tried out on drop.org itself as new additions to the software running the site.

It was only later, in January 2001, that Dries decided to release the software behind drop.org as "Drupal." The motivating factor was to enable others to use and extend the experimentation platform so that more people could explore new paths for development. The name Drupal, pronounced "droo-puh," is derived from the English pronunciation of the Dutch word "druppel" which stands for "drop."

Druplicon (the logo)

After Drupal had been created, an obvious matter was the choice and creation of a logo. Of course it would have to do something with a drop... or water.

The initial idea was simple: a drop in a circle. . It was featured as an "O" in a liquidish "Drop".

When the community grew, the idea came up of a cartoony drop with a face. Steven Wittens (UnConeD) created a 3D drop, but the idea didn't get too far mainly because 3D is hard to print, hard to edit, etc.

When the logo-issue had come up again, Kristjan Jansen (Kika) came up with idea of putting two side-way drops together to form an infinity-sign. When put into a filled circle, it resembled a face. After some more work by Steven Wittens, the Druplicon was created: a stylised drop with the infinity eyes, a round nose and a mischievous smile.

That's the 'story' behind it... I like the idea that the infinity-eyes symbolise the infinite possibilities that Drupal offers :)

See more versions of the logo in the marketing section.

Mission

By building on relevant standards and open source technologies, Drupal supports and enhances the potential of the Internet as a medium where diverse and geographically-separated individuals and groups can collectively produce, discuss, and share information and ideas. With a central interest in and focus on communities and collaboration, Drupal's flexibility allows the collaborative production of online information systems and communities.

Principles

- *Collaboration.* Drupal development supports open, collaborative information sharing systems and approaches (including systems such as community moderation of posts).
- *Standards-based.* Drupal supports established and emerging standards. Specific target standards include XHTML and CSS.
- *Open source.* Drupal is based on the open source philosophy of collaborative free software development and is licensed under the GPL. Drupal is itself open source and builds on and supports other open source projects. Specifically, Drupal is coded in the open source scripting language PHP and supports as primary data sources the open source database formats MySQL and Postgresql.
- *Quality coding.* High quality, elegant, documented code is a priority over roughed-in functionality.
- *Ease of use.* Drupal aims for a high standard of usability for developers, administrators, and users.
- *Modular and extensible.* Drupal aims to provide a slim, powerful core that can be readily extended through custom modules.
- *Low resource demands.* To ensure excellent performance, Drupal puts a premium

on low-profile coding (for example, minimizing database queries). Drupal should also have minimal, widely-available server-side software requirements. Specifically, Drupal should be fully operational on a server with Apache web server, PHP, and either MySQL or Postgresql.

Usability aims

For *developers* Drupal aims for a development system that is:

- *well-tooled* with a system of hooks that provide ready means to accomplish most foreseeable coding aims that involve interaction with core elements

For *administrators*, Drupal aims to provide solutions that are:

- *easy to install and set up* so that there is a minimal requirement for specific technical expertise
- *intuitive and self-explanatory* so that administrators can easily find the configuration options they need
- *highly configurable* so that site administrators can present just the interface they wish

For *users*, all elements of the Drupal user interface should be:

- *intuitive and self-explanatory* so that users with minimal prior experience can easily discover, navigate, and use functionality
- *uncluttered* so that users are not faced with a difficult task of sorting the essential from the non-essential

Presentations and articles

- Intranet Journal. Drupal: Powerful and Free, But Some Assembly Required - "*What Drupal does provide is an extensible framework, especially beneficial for use on larger intranets, which will allow you to expand and improve your intranet over time. The screens for adding new articles are simple, and the administrator of the system is given the ability to veto content submitted by contributing authors. If you have the time and expertise, it's well worth getting to grips with the nitty-gritty of Drupal if you'd like to fully customize its operation.*" [[read more](#)]
- The Fuzzy Group: performance of open source portal software - "*I have been a small part of the Open Source community since 1996 and I've been a regular Unix user since 1986. These technologies, which grew up on the Internet, offer compelling benefits for most organizations. A recent experience with an Open Source portal application, Drupal, pointed out to me just how good the performance of Open Source applications can be ? when it is done correctly.*" [[read more](#)]
- Teledynamics Communications: community plumbing for the web - "*Drupal is, as it claims, Community Plumbing, an infrastructure, a framework for building websites which serve a community of interest, but it's also more than this. Drupal has the latent ability to transform the web from a glut of brochures to a dynamic ecology of knowledge, a community record as much as it is a community forum.*" [[read more](#)]
- K-logging: supporting KM with web logs - "*There are many robust web log tools that are inexpensive or even free. Popular software includes MovableType, Radio Userland, any of the variations of Slashcode, and my favorite, Drupal. They allow individuals to publish content to a web site easily, and some packages even allow for categorization of entries. Most packages also permit*

authors to publish an XML feed of content. These low-cost tools help knowledge workers with two core concerns of KM: knowledge creation and knowledge sharing."

Community

Drupal is more than software - it is a project and a community. This section presents the structure and decision-making process in Drupal. There are various roles and responsibilities that people can assume in the Drupal project.

Development and support

As a communication center and project management space, drupal.org includes members who use Drupal as a personal website solution; IT professionals implementing Drupal for clients; and programmers, writers and others contributing to the growth of the Drupal open source project. Members work together to maintain extensive development and support resources on site:

- **Support.** Users experiencing difficulties installing and configuring Drupal should first consult the administrator's guide, much of which is also available through *help* in the *administration* section of every Drupal installation. In cases where documentation fails to provide a solution, search the support forum and drupal-support mailing list archives. If the solution is not available, please write a detailed description of the problem, include the Drupal version number, and post it to either venue. *Note:* all support is provided on a volunteer basis and is dependent on the good will of community members; please be patient with any support requests.
- **Development.** The Drupal developer's guide contains information on Drupal architecture, API specifications, guides for theme and module developers, and instructions for contributing your code to the project. The Bug tracker system should be used to submit bugs, ideas for new features, suggestions for improving drupal.org, and contributing ideas for usability and documentation. Those seriously interested in contributing to development should also consider joining the drupal-devel list.

Learn more

See the links below, the other sections of The Drupal Handbook, and the many discussions in the forums for more information.

The Drupal core

- *Founder and Lead Developer.* Drupal was founded by Dries Buytaert, who retains primary control over the software and makes most decisions on proposed changes. In approving or rejecting proposals and patches, he gives special weight to comments made by individuals whom he trusts and respects based on their past contributions to Drupal.
- *CVS review team.* A small team that reviews proposed changes and maintains code. They are the only ones who have write access to the core CVS repository. Current CVS review team members are Dries, Kjartan and Steven.
- *Maintainer.* While not directly making decisions, maintainers have informal responsibility

for a designated portion of the core (e.g., a particular core module). Individual areas of responsibility are listed in the file MAINTAINERS.txt. Maintainers are appointed by Dries. Core contributors who have made substantive contributions (particularly to a core component not individually maintained) may apply for Maintainer status by writing to Dries; Dries may also individually invite them.

- *Core contributor.* Core contributors are those who contribute code patches or documentation for the Drupal core, contributions that are peer reviewed and then decided on by Dries or other members of the CVS review team.

Contributions

- *Contributions repository manager.* The CVS repository of Drupal non-core "contributions" (mainly, modules and themes) has a maintainer, who reviews and approves applications for CVS access, and one or more other team members who fill in when the Maintainer is unavailable or otherwise occupied.
- *"Contributions" contributor.* "Contributions" contributors develop and maintain "contributed" code packages that are hosted on the Drupal site but not part of the Drupal core. A contributions contributor has applied for and received write access to the "contributions" CVS repository. Contributions contributors are improving the overall reach of Drupal by producing and sharing enhancements that can be used by others. Contributions contributors are generally listed in the README or CREDITS files included in module and theme downloads.

Documentation and support

- Documentation and support is collaboratively delivered by people in all Drupal roles, mainly through drupal.org and the development and documentation mailing list. Some drupal.org members have been granted rights to post and edit content and so directly author documentation like the Drupal Handbook.

Users

- *User.* Users are the people who use Drupal. Users don't generally contribute code but make valuable contributions by submitting bug reports or feature requests through the issues system (<http://drupal.org/project/issues>) and participating in the drupal.org forums (<http://drupal.org/forums>)

Download

The Drupal core platform, additional plug-in modules, and many theme templates are freely available for download under the GNU GPL. Drupal, written in PHP and using either MySQL or PostgreSQL as the database backend, can run on many platforms, including Apache or Microsoft IIS web servers.

More complete information and specific instructions about system requirements, installation and configuration are available in the administrator's guide.

Support

There are many ways to get support for your Drupal-based project or site.

- Browse the on-line documentation at <http://drupal.org/handbook>.
- Search the forum discussions at <http://drupal.org/forum>, and post questions.
- Keep up with the latest developments, ask questions and help to shape the development of Drupal by participating via the various Drupal e-mail lists. Sign up at <http://lists.drupal.org>
- Report an issue or request a feature via Drupal's issue tracker, online at <http://drupal.org/project/issues>
- Hire one of the many professional hosting and consulting/development services specializing in Drupal. A list of such services may be found at <http://drupal.org/services>

Hosting and services

The following subsections highlight people and organizations offering services related to Drupal. In addition to the listings provided on this page, there is a list of Drupal.org account holders who have indicated that they provide Drupal-related services. This list is automatically generated from user profiles and can be viewed at <http://drupal.org/profile/drupal-services>.

Drupal hosting

This section lists some organizations which specialize in hosting Drupal-based web-sites.

To be listed on this page: Send an e-mail to drupal-devel@drupal.org with the appropriate details. The current page maintainer will then add your organization to this page.

GrafiX Internet B.V.

GrafiX Internet B.V. provides transit, co-location, and dedicated servers in Amsterdam and Rotterdam, The Netherlands. We are most proud to be the dedicated server provider of choice for www.drupal.org, as well as some offspring projects such as www.drupaldevs.org.

We believe in 'medieval marketing', and thus our web presence (www.grafix.nl) is fairly humble. We strive to make our combination of service and support legendary, and our name to pass mouth to mouth, spread wide and far by our many satisfied customers.

It would honor us if you will consider GrafiX Internet B.V. as a service provider for your drupal-based deployment! Contact us at sales@grafix.nl or by phone at +31-(0)180 - 450170

We can offer:

- Server co-location starting from 59 €/month (Our network, your hardware).
- Dedicated Servers starting from 200 €/month (Our network, our hardware).
- Raw or managed transit capacity starting from 1 Mbps to gigabits per second.
- Rack (cabinet) space starting from 1/3 rack and up to entire datacenter cages.
- Network, operating system, and security consultancy.

Norrix

Norrix has launched their hosting service for Drupal. What this means is we will manually install Drupal for you, unlike other hosting companies. We will even help you get started on it, if its the first time you are using it. We will also **update** the software for you, **free of charge**, when a new version is released.

This is what you get for \$10 per month:

- Latest Drupal Installation
- 300 MB of webspace + 5GB Data Transfer per month
- 5 Email accounts
- 2 Databases
- 2 Mailing list
- And loads more...

For more information you can mail us at solutions[dot]norrix[dot]com. Also check the our web hosting page.

What more if you select us we will **donate** up to 50% of the profit to the development of Drupal. In this way you also help the evolution of the software and support the developers.

OpenSourceHost

OpenSourceHost is a specialized web hosting company focusing on providing quality web space and support for open source content management systems, as well as other open source software systems. For Drupal hosting, we provide graphical installation instructions, and if you take advantage of our special offer at <http://drupal.opensourcehost.com/> you will receive an additional 100 megs of space and 1 gig of bandwidth added to the hosting package of your choice.

CascadeHosting

A small webhosting company run from Portland Oregon, CascadeHosting offers cheap web hosting (\$99/year includes free domain registration) and web programming contract services. We'll setup Drupal for free as part of our \$99/year account, and answer any drupal related questions at drupal@cascadehosting.com. For more information, check their Drupal page.

Drupal services

This section lists some of the people and organizations providing Drupal-related services including design, installation, custom Drupal software development, training, support and consulting services.

We only list individuals and companies that have contributed to Drupal.

To be listed in the service directory: post an e-mail to Drupal developer mailing list (subscription required) with the appropriate details. The current page maintainer will then add your organization to the directory.

Moshe Weitzman

Contact Moshe. He lives in Boston, MA USA.

Services: Consulting on Drupal installation, training, and support. Custom Drupal software development also provided.

Moshe is intimate with Drupal's inner workings, and can complete custom projects with speed and quality. He authored much of the

- Distributed Authentication
- e-mail handling
- official Maintainer of Drupal's user system
- Hooks such as _head(), _exit(), and _syndication().
- glossary module
- syndication module
- taxonomy_dhtml module
- folksonomy module
- poor mans cron
- cooking recipe
- scheduler
- organic groups
- significant system documentation

Recent Clients

- Pixelworks is deploying Drupal in their intranet. They contracted with me to write an LDAP module, and an events module. Thanks Pixelworks.
- Moodcenter.org is deploying a portal site where patients complete surveys and receive instant graphical feedback about their mood state over time. This portal requires integration with a survey engine, statistics application, and PHP graphing utilities.
- Marlboro College is integrating the Drupal authentication system with their own LDAP based directory. The Drupal ldap_integration.module is powering that integration.
- National Society of Hispanic Professionals is relaunching their web site using Drupal as a Content Management System and community engine. Special planned enhancements

include a powerful new calendar with deep taxonomy integration.

- Music For America based their ambitious site on Drupal, and asked Moshe to develop modules for tracking their artists, venues, contacts, and more. Moshe delivered a flexible node module which could serve all these purposes at once. This module was incorporated into the Civicspace project. Planned enhancements include affiliate tracking and enhanced subscription features.
- University of Vienna is now running one of the most advanced Drupal pods in the world [staging site]. They maintain one Drupal site for many courses in their catalog, while maintaining a single user account across all sites. They also share language translations across sites. Moshe's design notes for this implementation are documented in this email (note: the stumbling block was solved).
- Rowland Institute at Harvard uses Drupal as an intranet for their community of scientists and technicians. Moshe delivered installation and webmaster training to Rowland, along with ongoing support.
- CodeOrange is a thriving community site based loosely on current news. Moshe developed the Node Moderation module which promotes the best nodes to the home page every day based on moderation ratings submitted by the community. Additionally, Moshe is currently working on enhancing Drupal's moderation systems in order to highlight strong posts, and hide troll posts.
- ShareNewYork is a community web site built around legal online sharing of music. It is a marketplace where users may upload songs and then receive commissions based on how many users download and purchase these songs. It is an innovative business model in a sector which has shown promise, but never made much money. Moshe is delivering custom modules for upload/download, automatic MP3 data extraction, FTP integration, ecommerce and PayPal integration, and more.
- Finnish Broadcasting Company enables their users to create and grow organic groups. These groups are similar to Yahoo Groups, where anyone can create a public or private group, and users post messages to their group home page. FBE has also sponsored Moshe to build photo gallery functionality based on a tagging system like Flickr and Del.icio.us. This work is being released back to the Drupal community. Thanks FBE.

Teledynamics Communications

Teledynamics Communications Inc is an internet and open-source consulting company based in Sauble Beach, Ontario Canada. Established in 1983, TCI has been involved in large-scale Internet portal research for over 10 years. Our portfolio includes community sites for military, manufacturing and emergency response applications, Sympatico-Lycos and the Canadian Broadcasting Corporation. For more information, visit Teledynamics Communications' Drupal services page or check their Drupal related information.

Gerhard Killesreiter

killes@drupaldevs.org
Freiburg, Germany

Gerhard is a freelance Drupal IT consultant. He has closely followed and participated in Drupal's development for over four years.

Services Consulting on Drupal setup and training, custom extensions to existing and development of new modules according to the client's specifications.

Qualifications During my work with Drupal I have implemented solutions for a variety of problems including - but not limited to - an access control module, a remindme extension for the event module, which I also maintain, and the listhandler module. In the past I managed to reduce Drupal's execution time by improvements to the database queries. Recently I have been successfully trying to decrease Drupal's page execution time even further by caching some data structures. I have also been successfull in getting a significantly improved locale.module into the Drupal core for the 4.5 release and worked on PHP 5 compatibility for the 4.6 release. A pending project is the improvement of the Drupal's revisioning system.

Recent Clients

- Familytimes.com required some improvements to the Amazon module.
- University of Vienna, Department of Pedagogics, required a module to evaluate their students' progress.
- Design Nine, Inc., project to be disclosed.
- TokenCoach, project to be disclosed.

Steven Wittens

steven@acko.net

Bonheiden/Leuven, Belgium

Services: Custom Drupal development (modules) and design (templates and themes). Contact me with your needs and specifics and we can work something out.

Qualifications: I am a long-time Drupal core developer, so I have intimate knowledge of the code and its features. Specifically, I have authored most of the filter system (which handles transforming the user-supplied text into HTML), several filtering modules (HTML Corrector, Smileys, URLfilter) and core's Poll module. I've also worked on making sure Drupal was Unicode/UTF-8 compatible. For Drupal 4.6 I have worked on improving the search.module.

I created two of the original Drupal themes as well as the contributed FriendsElectric theme. I run my own Drupal site, which has a fully validating and accessible XHTML/CSS theme. I also designed the theme for the Drupal.org website (Bluebeach).

Webschuur.com

webschuur.com is a small scaled company that builds content management system (CMS) driven websites. We can provide the help and advice to create a dynamic website, from scratch or from an existing site. Whether you are looking for cutting edge technology for your organizations web-based communication, or for solid solutions for your companies web-presence: we can offer it!

For more details please do not hesitate to get in touch with us.

Bèr Kessels (ber@webschuur.com)
Turnhoutsebaan 34/3
2140 Antwerpen
België
Telephone ++32 (0)3 6632292
www.webschuur.com

Trae McCombs and Kyle Smith

tmccombs@gotnerd.com, ksmith@gotnerd.com
Atlanta, Georgia USA / Reno, Nevada USA

gotnerd? is a Web Design & Development / Technology Services company that uses Drupal exclusively.

Services: We provide a one stop shop for all of your Web Development needs. Consulting, setup, training, custom extensions to existing and development of new modules according to the client's specifications are but a few of the things we can do.

Qualifications: We have built several big name websites: Two of them being Linux.com and themes.org. Should you be interested in seeing any of our work, or perhaps reviewing a quote from some of our clients, simply visit our website: <http://gotnerd.com/>

Heydon Consulting - Gordon Heydon

gordon@heydon.com.au
Melbourne, Australia

Services: Consulting, Drupal implementation, Customisation of modules and core to allow for a best intergration into your business. The creation of plugins for HTMLArea module to allow for better intergration, and not to disadvantage users who are not using HTMLArea.

Qualifications: Gordon has been an active member of the Drupal community since 2001, by first maintaining and contributing to existing modules. Later developing several modules such as the HTMLArea intergration module which allows for `<textarea />` tags to be converted into a WYSIWYG editor. Other contributed modules include the filestore2 module and an extensive modification to the image module. In addition to the contributed modules, their has also been a number of contributions to core in the form of new APIs and extensions of existing functionality.

Heydon Consulting is dedicated to the ongoing development of drupal, and working with customers to create a CMS that will fit your needs now and into the future.

For more information see <http://www.heydon.com.au/?q=consulting>

Matt Westgate

Contact Matt (aka mathias on drupal.org)
Ames, Iowa

Services: Custom Drupal development and project consulting.

Qualifications

I'm the author of the following modules, 3 of which are in the top 12 downloaded modules on Drupal.org.

- Official maintainer of the URL aliasing feature in Drupal core
- Ecommerce package
- Img_assist - An easy way insert/upload images into content
- TinyMCE rich-text editor integration
- Role-based node permissions
- Bookmarks
- Menu integration into the content form

Examples (portfolio)

- Entomology Index of Internet Resources - www.ent.iastate.edu/list
- Asian Soybean Rust - soybeanrust.info
- ISU Plant Pathology - www.plantpath.iastate.edu
- European corn borer biology - www.ent.iastate.edu/pest/cornborer

Károly Négyesi

Contact Károly
Budapest, Hungary

Services: everything PHP, mostly module and theme programming.

Qualifications: I know Drupal core quite well and I have submitted a patch in the beginning of 2005 which helped some Drupal pages to be up to 50% faster in a default-install scenario. Since then, my name can be regularly seen in the monthly Drupal quickies. I have helped creating i18n module and I am pretty familiar with creating multinational content sites. I lead the Drupal security team.

As I am from Hungary, a Second World country, my fees are usually lower than those from the First World countries.

2bits.com (Canada)

Contact 2bits.com

2bits.com is based in Canada, and offers web development using the powerful, flexible and open Drupal Content Management System and Framework for powerful database driven backends for web sites.

We offer consulting on module development, customization, installation, configuration, and maintenance, as well as hosting of your web site.

We have developed contributed modules such as:

- Feedback
- Stock
- Currency Exchange
- Google AdSense
- customererror
- SiteMenu

We also contribute to other areas within Drupal, and the Drupal.org web site.

Donating to the Drupal project

Donate

Drupal currently uses PayPal for receiving donations. Click the Paypal button below to donate money.



If you are inspired to donate something, but do not want to use PayPal, please see this excellent HOWTO on donating to Open Source projects. If you are willing to pay for particular enhancements, consider contacting someone listed on the services page.

Expenses

Your donation will be used to help the Drupal project. For example, by paying for:

- Hosting: server and bandwidth for drupal.org
- Development bounties
- Marketing material: flyers, posters, t-shirts, ...

Donors

If you donate money using PayPal, you are automatically added to the list of donors. Drupal will try to connect your PayPal and Drupal accounts using your e-mail address. If Drupal fails to connect both, update your drupal.org account so your e-mail address matches the one registered with Paypal.

The table below provides an overview of the people who donated money to Drupal. The numbers represent the net amounts after subtraction of transaction fees.

| Name | Amount | Date |
|--|----------|--------------|
| Richie Carey (Richie Carey) | € 3.63 | 20 hours ago |
| Scott Goodwin | 4.41 USD | 1 day ago |
| Caroline Clep (Digital Clouds Ltd.) | € 201.62 | 2 days ago |
| Fredrik Jonsson (Combonetwork development) | € 28.63 | 4 days ago |
| Adrian Russell-Falla | € 14.06 | 5 days ago |
| Jižnos Magyar | € 0.62 | 2 weeks ago |
| BRUNO LESSA | € 9.26 | 2 weeks ago |
| BRUNO LESSA | € 9.26 | 2 weeks ago |
| BRUNO LESSA | € 9.26 | 2 weeks ago |
| Jake Cole | € 39.5 | 2 weeks ago |
| Sam Raheb | € 9.26 | 3 weeks ago |
| Alissa Catalan ... | € 38.09 | 4 weeks ago |
| Scott Goodwin | 1.91 USD | 4 weeks ago |
| Fabio Varesano | € 38.29 | 5 weeks ago |
| Sam Raheb | € 3.49 | 5 weeks ago |
| Brady Jarvis (codeorange.net project) | € 95.75 | 5 weeks ago |
| John Cesario | € 36.89 | 5 weeks ago |
| Alan Nouri | € 9.26 | 7 weeks ago |
| Jan Visser (EyesOnSales) | € 47.7 | 7 weeks ago |
| Marco Laverdiere | € 95.75 | 7 weeks ago |
| Scott Goodwin | 0.56 USD | 8 weeks ago |

| | | |
|---------------------------------------|----------|--------------|
| Miguel Duarte (...) | € 18.97 | 9 weeks ago |
| Jeffrey Smith (...) | € 359.06 | 10 weeks ago |
| Mats Staugaard (xmac.no) | € 9.26 | 11 weeks ago |
| Linus Lee | € 18.87 | 12 weeks ago |
| Harald Walker S... | € 18.97 | 12 weeks ago |
| Yolanda Chiesa ... | € 5.42 | 12 weeks ago |
| Scott Goodwin | 0.56 USD | 13 weeks ago |
| Brady Jarvis (codeOrange.net project) | € 95.75 | 13 weeks ago |
| Jeffrey Smith (JCS/Monogram Holdings) | € 18.87 | 13 weeks ago |
| Jan Visser (EyesOnSales) | € 47.7 | 13 weeks ago |
| Jeffrey Smith (JCS/Monogram Holdings) | € 33.28 | 14 weeks ago |
| Scott Goodwin | 0.56 USD | 17 weeks ago |
| Kimio Yamakita | € 6.4 | 17 weeks ago |
| Craig Fifield | € 4.45 | 17 weeks ago |
| ANDRE-P GAGNON | € 4.45 | 17 weeks ago |
| Carlos Miranda Levy | € 23.67 | 17 weeks ago |
| Rob Stead | € 28.48 | 18 weeks ago |
| Omar Bickell | € 9.26 | 20 weeks ago |
| Ber Kessels | € 47.95 | 20 weeks ago |
| David Bartmess | € 9.26 | 21 weeks ago |
| John Hurley | € 9.26 | 21 weeks ago |
| Leonard Feldman | € 71.72 | 21 weeks ago |
| Todd Cochrane (...) | € 47.7 | 22 weeks ago |
| Fermí San Nicolas | € 23.67 | 22 weeks ago |
| Michael Karliner | € 18.87 | 22 weeks ago |
| Geoffrey White | € 9.26 | 22 weeks ago |
| Boris Mann | € 350 | 23 weeks ago |
| Kimio Yamakita | € 0.53 | 23 weeks ago |

| | | |
|---------------------------|---------|--------------|
| Kimio Yamakita | € 9.26 | 24 weeks ago |
| Hasan Yalcinkaya | € 18.87 | 24 weeks ago |
| Theodor S. Weinberg | € 4.45 | 25 weeks ago |
| James Walker | € 47.7 | 25 weeks ago |
| Zachary Rosen | € 23.67 | 25 weeks ago |
| Andrew Cohill (...) | € 23.67 | 25 weeks ago |
| Matthew Schwartz | € 287 | 26 weeks ago |
| Keith Instone | € 95 | 26 weeks ago |
| Alberto Luis Knapp Bjerer | € 95 | 27 weeks ago |
| Brady Jarvis | € 95 | 28 weeks ago |
| Community Publi... | € 33 | 28 weeks ago |
| Wayne Bartling | € 18 | 30 weeks ago |
| Michael Schalla | € 9 | 31 weeks ago |
| Morrissey-solo | € 47 | 31 weeks ago |
| Baba Buehler | € 18 | 32 weeks ago |
| Combonetwork development | € 28 | 32 weeks ago |
| Cielo Systems Inc. | € 18 | 32 weeks ago |
| Kath O'Donnell | € 47 | 32 weeks ago |
| Michael Heath | € 4 | 33 weeks ago |
| Ismael Fanlo | € 4 | 33 weeks ago |
| Evilsquid.net N... | € 9 | 33 weeks ago |
| Bryan Kennedy | € 16 | 33 weeks ago |
| Lynn Siprelle | € 9 | 33 weeks ago |
| Nick Berendsen | € 95 | 33 weeks ago |
| Jeremy Reichman | € 14 | 33 weeks ago |
| Eric Scouten | € 28 | 33 weeks ago |
| Moshe Weitzman | € 33 | 33 weeks ago |
| Progression Media | € 4 | 33 weeks ago |

| | | |
|--------------------|------|--------------|
| Karsten Müller | € 18 | 33 weeks ago |
| Papasoft | € 4 | 33 weeks ago |
| Neil Drumm | € 18 | 33 weeks ago |
| Bo Laurent | € 23 | 33 weeks ago |
| Lucas Koornneef | € 14 | 33 weeks ago |
| Chris Johnson | € 23 | 33 weeks ago |
| Progressive Val... | € 18 | 33 weeks ago |
| Webs4.com | € 9 | 33 weeks ago |
| Charles Lowe | € 18 | 33 weeks ago |

Newsletter advertising

Thanks for your interest in sponsoring the Drupal newsletter!

The Drupal newsletter is a monthly newsletter about Drupal usage and development. It's published via email or online on drupal.org. Archives can be seen here.

The Drupal newsletter advertising Program is an advertisement service offered to businesses in general interested in offering their products and services to Drupal newsletter subscribers.

Advantages

- Drupal newsletter subscribers are technology savvy and have an interest in internet technologies.
- Low placement fees.
- Income goes totally and directly to the Drupal project, so you are supporting Drupal development while at the same time marketing your products and services.

Ads format

An ad can have a maximum of 4 lines, each line a maximum of 70 characters, plus a link in a separate line. Links can also be included among the first four lines.

Space available

Up to two ads can be placed per Drupal newsletter issue, placed throughout the issue as deemed by the editor.

Fees

Placement fee is \$50USD per ad per issue. You may contract multiple placements at a discounted fee. All income generated by the advertising program goes directly to support the Drupal project.

Please contact us at newsletter-owner@drupal.org for more information on this service.

Version 4.6 roadmap

update: This roadmap was an experiment. IMO, the outcome of that experiment is not very good. I handedited this page quite often, but people promised much more than they could ever do. So not even half of the items are actually finished. For 4.7 we must either make a more general roadmap, or write software to maintain this roadmap better, so that developers are more motivated to actually do what they promised.

This is a table that shows "who is doing what" on our road to 4.6.

Using this table, we can group our efforts, and others can track what we are doing and how far we are.

Please try and keep this page up to date as much as possible. If you join or leave a team, edit this page. If a status changes, please edit this page too. If you have long descriptions and documents you would like to add, please add book pages under this chapter.

| Task | Modules/Areas | Team of volunteers | status |
|--|----------------------------|--|--------|
| Theme improvements <i>More information in the civicspace labs</i> | theme system; block system | Neil Drumm Chris Messina Halfelven Stefan Nagtegaal | WIP |
| Design themes <i>Design a total of ten 10 themes, templates or styles, of which at least 5 are templates or themes.</i> | themes | Adrinux Bèr Kessels Mega Grunt sepeck Stefan Nagtegaal Steven Wittens | WIP |
| End-user documentation | drupal.org; help hooks | Bert Boerland Bryght | Done |

| Task | Modules/Areas | Team of volunteers | status |
|---|-------------------------------|--|--------|
| Internationalisation i18n | i18n; locales | Jose A Reyero Károly Négyesi > > Carl McDade Bèr Kessels Adrian Rossouw | WIP |
| Translate interface <i>Have a total of 7 translations released for 4.5</i> | locale; po files | Bèr Kessels Gerhard Killesreiter Stefan Nagtegaal | Done |
| Content Construction Kit / metadata <i>more information</i> | flexinode; node | Jonathan Chaffer Bèr Kessels John VanDyk Neil Drumm Matt Westgate | WIP |
| Search improvements <i>Fix the search, according to</i> | search hooks; search module | Steven Wittens | Done |
| Install system <i>Introduce an install wizard system</i> | core; modules | Adrian Rossouw | WIP |
| Move to business area <i>Make Drupal able to act as backend for "leaflet" sites, purely corporate and mostly simple websites</i> | core; modules (drupalCOM) | Jose A Reyero Bèr Kessels | TODO |
| Move to project area <i>Make Drupal able to act as a groupware and project management tool</i> | project modules, groupware | Uwe Hermann dikini | TODO |
| Improve content organization <i>Introduce and improve modules to organise and manage content</i> | mindmap.module book.module | Gerhard Killesreiter Magico | TODO |

| Task | Modules/Areas | Team of volunteers | status |
|--|----------------------------|--|--------|
| Improve menu system | menu.inc; menu.module | Jonathan Chaffer | WIP |
| Improve and fine grain permission system <i>adding custom items to menus that are only visible to certain users.</i> | menu system; core; | Jonathan Chaffer | WIP |
| Improve block administration | block.module theme system | Neil Drumm sandip | Done |
| Fund-raising and marketing | drupal.org | Lapurd Bryght Bert Boerland | Done |
| Taxonomy system improvements <i>Taxonomy: standardize vocabulary metadata; open/closed vocabularies; interface to vocabularies in ways other than simply a selectbox Publish/Subscribe: share and aggregate vocabularies among Drupal sites</i> | taxonomy system | John VanDyk Mathias | TODO |
| RSS improvements | syndication system and API | Neil Drumm rkendall | Done |
| Image module improvements | image module, file.inc | Eric Scouten > > James Walker | Done |

User's guide

Welcome! This *User's Guide* describes how to get started with a Drupal-powered web-site. The guide covers basic topics such as registering for an account, logging in, changing your account settings, and creating content.

Drupal is a *content management system*. Its goal is to help users compose and present web-site content such as articles, photos, and other content types. Drupal is a "dynamic" system - rather than forcing users to specify a fixed, pre-declared arrangement of content, Drupal takes care of the details of how information is arranged and presented, and lets users focus on the actual content to be displayed.

Most of the content on a Drupal-based site - the text of this page, for example - is stored in a database. Text and images are submitted by filling in forms via a web-browser. When visitors view a page, Drupal gets the relevant bits of content from the database and composes all of the components of the page in a template. This makes it easy to quickly add or change content, without requiring knowledge of HTML or other web-technologies on the part of the person providing the content.

Depending on the configuration of the Drupal site and the user-roles you play with respect to that site, you may be allowed to contribute or edit content. Fortunately, Drupal is designed to make this relatively easy. Very little technical knowledge is assumed. Though details may vary with a site's configuration, the basic process involves these steps:

- *register* with the site
- *log in* by typing the user name and password supplied you in the registration step, and
- *create content* (e.g., articles, stories) into forms.

This user guide will explain these steps and familiarize you with the basic information you need to use Drupal successfully.

Registering as a user

To add or edit content on a Drupal site, usually you have to first be registered as a user. (Sometimes the site administrator has chosen to enable "anonymous" posts of things like comments, in which case you can post them without registering.)

In some cases, a site administrator will add you as a user. If so, they will send you a user name and password that you can use to log on.

Otherwise, look for a small form called "User login" on the main page of the site you want to register with (usually on the right or the left of the page). Click the link that says "Create new account".

The next page that comes up will generally have some information on the site's policies for registration. After reading them, to register, enter a user name of your choice and an email address to which you have access and hit "submit". Then check your email account. Within a few minutes, you should get an automatically-generated email confirming your registration and giving you an initial password to use. Now you're ready to log in.

Logging in

Before you can add or edit content, you usually need to log in. If you haven't already done so, register as a user, see above (or, if applicable, request that your site administrator register you). Then hit the main page of the site you're wishing to use and look for a "User login" form. This will typically be on the left or right side of the page (it is a "block" in Drupal talk). Enter your user name and password and hit "submit".

Assuming everything's working as planned, when the new page loads it will include a new block with your user name at the top. This is the menu you use to start entering and editing content.

Changing your account settings

Once you have registered with a Drupal-based site, you can change settings to control information about yourself and also your use and experience of a Drupal site. To see what tweaks you can make to your account, log in and then click on *my account* in the navigation block (that's the one titled with your user name). Click on the *edit* tab.

Account Settings. You may see a different collection of settings than is presented here, depending on what features have been enabled on your site.

password

Enter in a new password in both fields to set it. Drupal sends you a default password that is often hard to remember, so it is recommended that you change your password to something you can easily remember.

block configuration

The site administrator may make some blocks (chunks of content that are usually displayed in a left and/or right column) optional. You can enable and disable the display of these blocks by checking and unchecking the boxes next to them.

signature

If comments are enabled, you will be able to set a default signature. This will be copied into new comments for you automatically, but may still be edited.

time zone

Your site administrator may allow users to set their time zone. This will cause all dated content on the site to display in local time, according to the offset you enter here.

theme

A "theme" is the basic look and feel of a Drupal site. Sometimes a particular site will have more than one theme installed. If the site administrator has made more than one theme available, you will be able to select what you would like the default theme to be for your

account.

As mentioned earlier, different site-settings will cause different fields to be displayed on your user account page. See the documentation for individual modules for instructions on how to use these additional options.

Additional Information. Aside from the account settings tab, you may also see additional tabs, titled according to the information they contain. Some examples might include "Personal Information", "Workplace", etc. These are controlled by the profile module, and allows you to enter more information about yourself. Please see the profile module for more information on this.

Creating new content

Once you have logged-in, you're ready to start posting content.

At the top of your personal menu, you'll find a link called "create content". Click this and you'll see a list of the types of content you can create. This list reflects the privileges assigned to your user account or to the group ("role") your account is part of.

There are several contributed modules which can assist with more complex content creation within this framework, such as spellchecking, image embedding, and file attachment uploading.

A step-by-step example

We will assume that you have selected *create content* and chosen "story" as a content type.

You should be looking at a form with the title "Submit story". From here, it is just a matter of filling in the form and posting it.

Administrative options

At the top of the form you may see some administrative options. For example, there is a box with the heading *User comments*. Drupal supports discussion/comments on postings--but such comments are not always appropriate. If your article is one that could be usefully commented on, keep the default settings: "Read/write". Otherwise, choose "Disabled".

Title

The title is straightforward enough. Try to be descriptive and catchy.

Topic

Next comes the "Topic" menu. This is the section your article will go in--or in the technical language of Drupal, its ("taxonomy categories"). This list presents all the sections available on the website, with their structure. So, choose the appropriate section or sections for your story and continue down the form to supply the body of your text.

Body

The "body" field is where you put the main content of the page. If you've typed this into a word processor or HTML editor, just copy and paste it into this field. Alternately you can just type straight in. For the most basic page, just type and leave a blank line (i.e., hit "enter" twice) at the end of each paragraph.

You can optionally format your entry in friendly old HTML. But hey, if you're a novice, don't worry--it's not as difficult as it sounds. Here's a quick primer:

If you want something to be **bold**, just enclose it in `` or `` tags, like this:

```
<b>This text is bold</b>  
<strong>This text is bold</strong>
```

Note that there is always an opening tag (no forward slash) and a closing tag (a forward slash before the tag name, indicating that you are "turning it off").

To make something *italic*, put it in `<i>` or `` tags:

```
<i>This is in italics</i>  
<em>This is emphasized</em>
```

There is considerable debate about the semantic nature of the `` and `` tags versus the `<i>` and `` tags.

To put things nicely in paragraphs, enclose them in `<p>` tags:

```
<p>This is a paragraph.</p>
```

To make a bulleted list, first open a list with a `` tag (that stands for "unordered list"), then put each list item in `` (yes, for "list") tags. Don't forget at the end to close off your list with a closing `` tag. Here's how it looks:

```
<ul>  
<li>This is the first bulleted item</li>  
<li>This is the second bulleted item</li>  
</ul>
```

The result is displayed below:

- This is the first bulleted item
- This is the second bulleted item

That wasn't too painful, was it?

Decide where you want the "teaser" (the part of the main text used in links to the article) to end. If you do nothing, Drupal will choose a breaking point for you--but it's better to decide yourself, to make sure the breaking point is appropriate. You do this by typing in a <!--break--> where the teaser is to end.

And you're set! You can preview the page you've prepared by hitting "Preview" (recommended, and sometimes required) or you can bravely or recklessly just go ahead and publish it by hitting "Submit".

Types of content

There are various types of content that you can post using Drupal. Many of these are organized into what are called "nodes". Basically, you can think of a node as the content of a page. This might be, for instance, an article. Content is added or updated through web page forms. So to add an article, you bring up a form, enter text into it (like the title and content of an article), and hit a button to submit the form.

Topics, categories and terms

Content on Drupal websites is usually organized using categories through a system called "taxonomy". A taxonomy has different "terms" that are used as categories for articles. When you're adding an article, you might find a drop-down list of topics. By selecting one, you choose where on the site to categorize your article. If this seems hard to relate to, you can think of topics as being like folders on your hard drive--they help to organize content, so that you can find similar things in the same place.

Permissions

What types of content you can create or edit depends on the privileges assigned to the "role" or user group that you're a member of. In general, to find out what you can do:

- On your user menu (the collection of links that has your user name as a title), look for a link that says "create content". Click this to get a listing of the types of content you have permission to post.
- On a particular page, look for links at the bottom of an article. These links say things like "12 comments" (if there are comments that have been made on the article) and "read more" (if you're looking at a short version of an article). If one of these links say "administer" or something like "edit this page", you have permissions to edit that type of content.

Moderation and the submission queue

Some Drupal sites are set up so that when you submit content it goes into a "submission queue". Content in this queue is read by other users who have a *moderator* role on the site. They will review the content and, if it is accepted, "publish" your contribution.

Creating comments

Comments allow users to interact with the content on a site, to respond to an article, offer their own ideas, make additions, or supply a critique.

Leaving comments

When you bring up an article to read, look for comment-related links at the bottom of the article. If you're not logged in, this might read "login or register to post comments". When you do log in, you should see something like "Add new comment". Click on the link and you're ready to comment away.

Etiquette

Comments can be a great way of enriching a community site--but they can also lead to unfriendly, even harassing exchanges. As with any communication, it's important to try to ensure that your comments are respectful and constructive.

"Threaded" comments

Comments on a Drupal web-site are "threaded". This means you can comment directly on an article--or you can reply to an existing comment. If you reply, your comment will be indented to show that it is part of that discussion.

Alternative ways to enter content

Drupal's built-in form-based content editing approach is fine for many applications, but if you have a lot of text to create, or you wish to convert existing content, or if you are using a specialized content type such as a blog, it may be more convenient to use other approaches to enter content into your Drupal-based site.

Preparing your content offline suggests some ways to use familiar software on your computer to create or edit content before submitting it to Drupal.

Depending on what's available on your site, you might even be able to enter new articles without ever logging on to the site.

Drupal includes functionality for "blogging"--creating "blogs" or web-based journals. If this functionality is enabled on your site, you may be able to input and edit content using one of a number of desktop "blog" software packages. These allow you to simply type in content, hit a "post" button, and have your content automatically loaded onto your site.

In fact, blogging software can be used for more than blogs--it can allow you to post content easily and quickly to almost any part of a website. One such program is described below in *Posting and editing content with w.bloggar*

Before trying out one of the blogging programs, you might want to check with your site's maintainer to make sure it accepts blog posts. The key question to ask is: "Is the bloggerapi enabled?" If the answer is yes, you're ready to roll. If not, you could request that it be enabled to allow you quick updates.

Posting and editing content with w.bloggar

w.bloggar is a gratis software for Windows designed for "blogs" (web-based journals).

If you've confirmed that blog support is enabled, here's some steps to get going:

- Download the software from <http://www.wbloggar.com/> and install.
- Set up a new account. This is explained in the w.bloggar help files.
 - When it comes time to set the "Blog Tool" selection, choose "MovableType" (and *not* "Drupal"). This is because (at time of writing) the Drupal support in w.bloggar is outdated.
 - For "Host" put the domain of the website you're using, then for "Path" put the rest of the address, if any, followed by "/xmlrpc.php". So if the address was "http://www.gworks.ca/site/" you would put "www.gworks.ca" for host and "site/xmlrpc.php" for Path. The "xmlrpc.php" part is the Drupal file that handles the blogging input.

Now you're ready to start posting. In doing so, you can take advantage of the text formatting functionality w.bloggar offers. When correctly set up, posting a web page from w.bloggar is as simple as opening the program, typing in some text, selecting a category (the "taxonomy term" to use) and hitting post.

Preparing content offline

Before posting directly to a site, you may want to start in a word processing program. Potential advantages include:

- Saving time online. This is a particular consideration if you're on dial-up.
- Access to spell-check and other editing features.

Depending on how much formatting you wish to do, you could also consider using an HTML editor. These include, for instance, the "composer" that comes with Mozilla and Netscape. Here are the steps involved:

- Type or copy and paste your text into the HTML editor.
- Apply formatting as desired (e.g., bold, italics).
- Bring up the HTML (encoded) view of the text.

This HTML is what you'll copy and paste into Drupal's input form, to have formatted copy.

Editing and deleting content

To edit or delete existing content, log in and then bring up the page you wish to edit. Look on the page for an "edit" tab. Depending on your user permissions, you might see this on all pages or only on certain ones (e.g., those that you yourself submitted).

Clicking the edit tab will bring up a page with a form for changing the page. Here you can change the text and settings. Once you have the text and settings in a suitable form, click on the "Submit" button on the bottom of the form. Note that certain sites may be set up to require you to "Preview" the page before you can submit your changes.

If you wish to delete the page (and you have appropriate permissions), click on the "delete" button near the bottom of the form. You'll get a second chance to confirm that you wish to delete the page--or to change your mind!

Note: Because Drupal is very configurable, there may be additional ways of editing and managing content. Please check the documentation for your installation, ask the Drupal administrator, or consult with another user for details.

Beyond the basics

Drupal is designed to support many different types of website. Many changes to a Drupal site's functionality, appearance, and modes of interaction are easy to make via Drupal's configuration and extension mechanisms.

- Drupal is highly *configurable*. The administrator of a site can enable different capabilities and change many settings that affect the look and functionality of a site.
- Drupal has a system of *privileges* that makes it possible to create different user *roles* - for instance, member, staff, partner. Each type of user can see and do different things on the site.
- Drupal is designed to be easily extended through *modules*--blocks of code that provide extra functionality or enhancements. Some modules come with every Drupal installation ("core" modules), while others can be individually downloaded and installed from the Drupal website ("contributed" modules).
- The look and feel of a Drupal site can be changed through different "*themes*". As with modules, there are both core and contributed themes.

Hence, what you see on a particular Drupal site, and what you can do there, depends to a high degree on what the site administrator(s) have chosen to present. If you require more in-depth information about changing the way in which information is presented or how its appearance may be configured, please see the *Administrator's guide* at <http://drupal.org/handbook> and the Drupal forums.

Administrator's guide

An administrator's guide for installing and configuring a Drupal site. This guide includes extensive How-to's for using all core modules.

Introduction

Drupal is a web-based *content management system*. Text, images, and other kinds of content are stored in a database, dynamically retrieved and composed, and presented to a user in response to a request sent via a web-browser. The details of how all of this happens -- and how you, the administrator of a Drupal site, can control how information is stored, retrieved, and presented to the user -- is the subject of this manual.

Terminology

Drupal uses certain terms to mean specific things. The fundamental elements of Drupal are defined below.

Block

Blocks are the navigational or content additions that live on the left or right side of a page when you view it in your browser. Blocks are not nodes, they are just a way of positioning data within a page. The look of blocks can be controlled by each theme by defining the `block($subject, $content, $region = "main")` method.

Configuring and Managing Blocks

Box

Box is a container for content on Drupal pages. Each box has a title and some content. The look of boxes can be controlled by each theme by defining the `box($subject, $content, $region = "main")` method.

Engine

A special type of theme that moves the HTML markup generation to template files (using any templating system). Also tells the theme selector what templates have been defined. Additional theme engines (`xtemplate` is the current theme engine included with core) are available from the theme engines section of downloads.

Filter

Framework for handling filtering of content.

Module

A module is a piece of code which extends Drupal to provide a specific piece of functionality. Some modules are part of the core Drupal system (eg. the taxonomy and blog modules) and some others (eg. the weblinks and image modules). Core modules are those included with the main download of Drupal. Contributed (or "contrib") modules are available for separate download from the modules section of downloads. Be sure that the version of the contrib module you wish to use matches your version of Drupal. The releases

section lists modules by Drupal version.

Node

Nodes are probably the hardest Drupal concept to grasp but they are really quite simple. Almost all content in Drupal is stored as a node. When people refer to "a node" all they mean is a piece of content within Drupal, it could be a poll, a story, a book page an image etc.

Permissions

Permissions control access to content creation, modification and site administration. Administrators assign permissions to roles, then assign roles to users. The first user of a Drupal site automatically receives all permissions, no matter what role that user belongs to.

Roles

Roles are groups with certain permissions that can be applied to individual users. Users can belong to more than one role. Two roles, authenticated user (those users that sign up for an account) and anonymous users (those either without an account or not logged in) are the default roles of Drupal installations, but they can be configured and the first user can create additional roles.

Style

A CSS file (or files) replacing the default CSS of a theme or engine. Appears in the theme selection list with the same precedence as themes and templates.

Taxonomy

Taxonomy is literally "the science of classification". Drupal uses taxonomy to describe the category system, which you can use to classify and organize content on your web site. In Drupal a taxonomy is a set of categories. There is additional information on the taxonomy system in the documentation.

Template

A HTML-writer-readable file that is mostly HTML with special codes to substitute in values provided by a engine.

Theme

A PHP file of functions which turn arguments into HTML markup. Drupal modules define themeable functions which can be overridden by the theme file. There are additional themes available in the themes section of downloads.

Node types

Drupal stores all of its content in *nodes*. Drupal's basic set of node types is relatively short, but quite flexible.

Blog Entry

Blogs, or weblogs, are another term for an online journal or diary. They are a place where members of the community can write their own thoughts and not have to worry about being ontopic for the site.

Book Page

Book pages are designed to be part of a collaborative book. An example of a collaborative book is the Drupal developer documentation. Originally only book pages could be a part of a book but these days all node types can be part of a book. Really the only special part about book pages these days is that like static pages they can contain PHP code.

Comment

Comments actually aren't nodes, they are their own special content type. Comments are what allow people to add comments to any other node that has been created.

Forum

Forums are the same thing as online bulletin boards. New forums can only be created by administrators of the site and are generally dedicated to a particular topic or question. Once a forum is created anyone can ask questions or comment on other peoples questions.

Page

Site pages are static pages which are typically (but not required to be) linked into the main navigation bar. One special thing about them is that they can contain customized PHP code in order to make their content dynamic.

Poll

A poll is where a multiple choice question is asked and users can answer and see other peoples answers to questions.

Story

Story pages are the generic page type that most content management systems have. Stories are generally used for information which is only relevant for a period of time (eg. news stories) and is expected to expire off of the page.

Additional types of nodes are provided by contributed modules.

Installation

System requirements

1. A Web Server that can execute PHP scripts
 - Recommended: Apache. Development with version 1.3.x. Successfully tested with version 2.0.x.
 - Optional: IIS. Drupal is being developed with IIS compatibility in mind, and IIS is reported to be working.
2. PHP
 - As of Drupal 4.6, the CMS requires PHP version 4.3.3+ (PHP 5 is supported for the 4.6 release). Drupal 4.2 to 4.5.2 inclusive require PHP version 4.1+. Older versions of Drupal will run on PHP 4.0.6+. We recommend using the latest version of PHP 4.x.
 - PHP XML extension (for {bloggerapi | drupal | jabber | ping}.module). This extension is enabled by default in a standard PHP installation; the windows version of PHP has

built in support for this extension.

- PHP needs the following configuration directives for drupal to work:
 - session.save_handler: user
 - In addition, we recommend the following settings:
 - session.cache_limiter: none
 - (we only mention directives that differ from the default php.ini-dist / php.ini-recommended starting with PHP 4.0.6)
- These settings are contained in the default .htaccess that ships with drupal, so you shouldn't need to set them explicitly. Note, however, that setting php configuration options from .htaccess only works
 - with Apache (or a compatible webserver),
 - if the .htaccess is actually read, ie. AllowOverride is not None,
 - if php is installed as an Apache module.
- See here for how to change configuration settings for other interfaces to PHP.
- Using a PEAR supported Database (see below) requires (of course) PEAR to be installed.

3. A PHP-supported Database Server

- Recommended: MySQL, v3.23.17 or newer (for our use of INNER JOIN's with join_condition's). MySQL 4 is fine.
- Optional: Any PEAR supported Database. Currently, only PostgreSQL is actively maintained and supported, though. Experiences with other Databases are greatly welcome.

Installing Drupal

```
// $Id: INSTALL.txt,v 1.20 2005/04/23 05:07:08 uncoed Exp $
CONTENTS OF THIS FILE
-----
* Requirements
* Optional requirements
* Installation
  - Drupal administration
  - Customizing your theme(s)
* Upgrading
* More Information
REQUIREMENTS
-----
Drupal requires a web server, PHP4 (4.3.3 or greater) or PHP5
(http://www.php.net/) and either MySQL (http://www.mysql.com/)
or PostgreSQL (http://www.postgresql.org/).
NOTE: the Apache web server and MySQL database are strongly recommended;
other web server and database combinations such as IIS and PostgreSQL
are possible but tested to a lesser extent.
OPTIONAL REQUIREMENTS
-----
- To use XML-based services such as the Blogger API, Jabber, RSS
syndication, you will need PHP's XML extension. This extension is
enabled by default in standard PHP4 installations.
- If you want support for clean URLs, you'll need mod_rewrite and
the ability to use local .htaccess files. (More information can
be found in the Drupal handbook on drupal.org.)
INSTALLATION
-----
```

1. DOWNLOAD DRUPAL

You can obtain the latest Drupal release from <http://drupal.org/>. The files are in .tar.gz format and can be extracted using most compression tools. On a typical Unix command line, use:

```
wget http://drupal.org/files/projects/drupal-x.x.x.tar.gz
tar -zxvf drupal-x.x.x.tar.gz
```

This will create a new directory drupal-x.x.x/ containing all Drupal files and directories. Move the contents of that directory into a directory within your web server's document root or your public HTML directory:

```
mv drupal-x.x.x/* drupal-x.x.x/.htaccess /var/www/html
```

2. CREATE THE DRUPAL DATABASE

This step is only necessary if you don't already have a database set-up (e.g. by your host). If you control your databases through a web-based control panel, check its documentation for creating databases, as the following instructions are for the command-line only.

These instructions are for MySQL. If you are using another database, check the database documentation. In the following examples, 'dba_user' is an example MySQL user which has the CREATE and GRANT privileges. Use the appropriate user name for your system.

First, you must create a new database for your Drupal site (here, 'drupal' is the name of the new database):

```
mysqladmin -u dba_user -p create drupal
```

MySQL will prompt for the 'dba_user' database password and then create the initial database files. Next you must login and set the access database rights:

```
mysql -u dba_user -p
```

Again, you will be asked for the 'dba_user' database password.

At the MySQL prompt, enter following command:

```
GRANT ALL PRIVILEGES ON drupal.*
```

```
TO nobody@localhost IDENTIFIED BY 'password';
```

where

'drupal' is the name of your database

'nobody@localhost' is the username of your webserver MySQL account

'password' is the password required to log in as the MySQL user

If successful, MySQL will reply with:

```
Query OK, 0 rows affected
```

To activate the new permissions you must enter the command:

```
flush privileges;
```

and then enter '\q' to exit MySQL.

3. LOAD THE DRUPAL DATABASE SCHEME

Once you have a database, you must load the required tables into it.

If you use a web-based control panel, you should be able to upload the file 'database.mysql' from Drupal's 'database' directory and run it directly as SQL commands.

From the command line, use (again, replacing 'nobody' and 'drupal' with your MySQL username and name of your database):

```
mysql -u nobody -p drupal < database/database.mysql
```

4. CONNECTING DRUPAL

The default configuration can be found in the 'sites/default/settings.php' file within your Drupal installation.

Before you can run Drupal, you must set the database URL and the base URL to the web site. Open the configuration file and edit the \$db_url line to match the database defined in the previous steps:

```
$db_url = "mysql://username:password@localhost/database";
```

where 'username', 'password', 'localhost' and 'database' are the username, password, host and database name for your set up.

Set \$base_url to match the address to your Drupal site:

```
$base_url = "http://www.example.com";
```

In addition, a single Drupal installation can host several Drupal-powered sites, each with its own individual configuration.

If you don't need multiple Drupal sites, skip to the next section.

Additional site configurations are created in subdirectories within the 'sites' directory. Each subdirectory must have a 'settings.php'

file which specifies the configuration settings. The easiest way to create additional sites is to copy the 'default' directory and modify the 'settings.php' file as appropriate. The new directory name is constructed from the site's URL. The configuration for www.example.com could be in 'sites/example.com/settings.php' (note that 'www.' should be omitted if users can access your site at http://example.com/). Sites do not each have to have a different domain. You can use subdomains and subdirectories for Drupal sites also. For example, example.com, sub.example.com, and sub.example.com/site3 can all be defined as independent Drupal sites. The setup for a configuration such as this would look like the following:

```
sites/default/settings.php  
sites/example.com/settings.php  
sites/sub.example.com/settings.php  
sites/sub.example.com.site3/settings.php
```

When searching for a site configuration (for example www.sub.example.com/site3), Drupal will search for configuration files in the following order, using the first configuration it finds:

```
sites/www.sub.example.com.site3/settings.php  
sites/sub.example.com.site3/settings.php  
sites/example.com.site3/settings.php  
sites/www.sub.example.com/settings.php  
sites/sub.example.com/settings.php  
sites/example.com/settings.php  
sites/default/settings.php
```

Each site configuration can have its own site-specific modules and themes that will be made available in addition to those installed in the standard 'modules' and 'themes' directories. To use site-specific modules or themes, simply create a 'modules' or 'themes' directory within the site configuration directory. For example, if sub.example.dom has a custom theme and a custom module that should not be accessible to other sites, the setup would look like this:

```
sites/sub.example.com/:  
  settings.php  
  themes/custom_theme  
  modules/custom_module
```

NOTE: for more information about multiple virtual hosts or the configuration settings, consult the Drupal handbook at drupal.org.

5. CONFIGURE DRUPAL

You should consider creating a "files" subdirectory in your Drupal installation directory. This subdirectory stores files such as custom logos, user avatars, and other media associated with your new site. The sub-directory requires "read and write" permission by the Drupal server process. You can change the name of this subdirectory at "Administer > Settings > File system settings". You can now launch your browser and point it to your Drupal site. Create an account and login. The first account will automatically become the main administrator account with total control.

6. CRON TASKS

Many Drupal modules (such as the search functionality) have periodic tasks that must be triggered by a cron job. To activate these tasks, call the cron page by visiting http://www.example.com/cron.php -- this will pass control to the modules and the modules will decide if and what they must do.

Most systems support the crontab utility for scheduling tasks like this. The following example crontab line will activate the cron tasks automatically on the hour:

```
0 * * * * wget -O - -q http://www.example.com/cron.php
```

More information about the cron scripts are available in the admin help pages and in the Drupal handbook at drupal.org. Example scripts can be found in the scripts/ directory.

DRUPAL ADMINISTRATION

Upon a new installation, your Drupal website defaults to a very basic configuration with only a few active modules, one theme, and no user access rights.

Use your administration panel to enable and configure services. For example, set some general settings for your site with "Administer > Settings". Enable modules via "Administer > Modules". User permissions can be set with "Administer > Users > Configure > Permissions".

For more information on configuration options, read through the instructions which accompany the different configuration settings and consult the various help pages available in the administration panel.

Community-contributed modules and themes are available at <http://drupal.org/>.

CUSTOMIZING YOUR THEME(S)

Now that your server is running, you will want to customize the look of your site. Several sample themes are included in the Drupal installation and more can be downloaded from drupal.org.

Customizing each theme depends on the theme engine. In general, each theme contains a PHP file themename.theme which defines a function header() that can be changed to reference your own logos.

Most themes also contain stylesheets to tune the colors and layouts; check the themes/ directory for READMEs describing each alternate theme.

UPGRADING

1. Backup your database and Drupal directory - especially your configuration file in 'sites/default/settings.php'.
2. Log on as the user with user ID 1.
3. Remove all the old Drupal files then unpack the new Drupal files into the directory that you run Drupal from.
4. Modify the new configuration file to make sure it has the latest and correct information.
5. Run update.php by visiting <http://www.example.com/update.php>.

MORE INFORMATION

For platform specific configuration issues and other installation and administration assistance, please consult the Drupal handbook at <http://drupal.org/>. You can also find support at the Drupal support forum or through the Drupal mailing lists.

Easy way to install drupal with the cPanel control panel

Here is how I use cpanel to help with drupal installation:

- Install the drupal distribution files
 1. Upload the drupal distribution file (e.g. drupal-4.6.1.tar.gz) into the server using cPanel's file manager.
 2. Extract the file, using cPanel's file manager.
 3. (optional) Move the extracted files and directories, one by one, so that all files will reside under the root public_html folder of your site. This optional step is much easier to do if you have a shell access. But there are 10 files/folders in the top level, so it's doable.
- Define the database
 1. Open CPanel.
 2. Open the "Mysql databases" screen.
 3. Choose a name to the new DB and press the "Add DB" button. I choose short names, e.g. "db1", since they are prefixed by my cpanel account name anyway. If my cpanel account name is levavie, the full db

- name will be levavie_db1.
4. Create a new DB user. Chose a name for the new user (e.g. "u1") and a password (e.g. "234"), and press the "Add User" button. The full db user name will be prefixed by my cpanel account name. If my cpanel account name is "levavie", the full db user name will be levavie_u1.
 5. Add the user as a permitted user to the database. Select the user, select the DB, then press the "Add user to db" button.
- Change the setting in settings.php: \$db_url = "mysql://levavie_u1:234@localhost/levavie_db1";
 - Upload the database.
 1. Open phpMyAdmin (there is a link in the button of the cpanel MySQL account maintenance screen).
 2. Select the newly created database.
 3. Open the SQL tab.
 4. Extract database.sql into a file on your LOCAL machine.
 5. Run the SQL script needed to create the drupal tables by uploading the file from your local machine.
 - That's all! Your drupal database is now setup and connected. You should point your browser to the top of your site (e.g. <http://www.example.com>) and start configuring it.

All the best,

Amnon

-
Web Architect, e-Collaboration Consultant
My Homesite (drupal Hebrew) | hitech-dolphin.com |
small-business-web-hosting-strategies.com

General instructions

Here is the procedure for installing drupal on a Linux or Unix system. This chapter describes the generic installation procedure for drupal as well as detailing some installation instructions for specific configurations.

1. Download the distribution tar-ball and unzip it into the directory you want to serve web files from:
 - `tar -zxvf drupal-x.x.x.tar.gz`
2. Create a MySQL database for your drupal site (if you haven't already):
 - `mysqladmin create database_drupal`
3. Create a user for your MySQL database and assign it the proper permissions.
 - **At the command line** (requires root access or permissions to create users) type the following lines. The first line runs MySQL, then the second line instruct the MySQL program to create the proper permissions.
 - `mysql -u root -prootpassword database_drupal`
 - `grant all privileges on database_drupal.* to username@localhost`

- identified by 'userpassword';
- **Important:** in the above examples, replace *rootpassword* with the root MySQL user's password, *database_drupal* with the name of the database you are creating, *username* the Drupal database's username, *userpassword* with the new password you are assigning to that MySQL user.
4. Once you have a proper database, dump the required tables into your database:
 - mysql -u *username* -p*userpassword* *database_drupal* <
database/database.mysql
 5. Edit the includes/conf.php configuration file to set the required settings such as the database options and to customize your site. You will need to know your MySQL username, password, and database name (see above).
 6. Launch your browser and point it to <http://yourdomain.com/> and create an account, log in. The first user will automatically have all administrator permissions. Click *my account* to edit your password.
 7. (Optional) Edit the .htaccess file and set the values of the PHP variables to your likings: session.name, session.cookie_lifetime, session.gc_maxlifetime, session.cache_expire and session.save_path. Check your PHP reference manual for the exact purpose of each variable mentioned.
 8. (Optional) Setup a crontab to periodically visit <http://yourdomain.com/cron.php>.
 - This usually means editing the /etc/crontab file and inserting a line like one (but not both!) of the following:
 - 00 * * * * /usr/bin/lynx -source <http://yourdomain.com/cron.php>
 - 00 * * * * /usr/bin/wget -O /dev/null
<http://yourdomain.com/cron.php>

Notes:

PHP.ini should have the following settings:

- register_globals=on
- allow_call_time_pass_reference = On

Special cases

Various special situations (e.g., installing Drupal behind a firewall, installing Drupal in a subdirectory of the webserver's document root) are discussed in this section. Information specific to particular operating systems or database platforms may be found elsewhere.

Installing Drupal behind an Actiontec GT701-WG router

The popular Actiontec GT701-WG (and possibly other routers) DSL modem/router has a peculiar behavior that affects how *base_url* must be set up in sites/default/settings.php. The solution is to replace the PHP code that sets *base_url* with some code that is available below.

Here's the scenario: you are using computer A to access and use your Drupal site, computer B is the server that is hosting the Drupal site, and both computers are plugged into a GT701 router, which in turn is linked to the Internet. Internet users can access your server from the outside by typing in www.example.com (because you set up port forwarding on the GT701). However, the GT701 will not allow you to type that address in from computer A, because the GT701 interprets such an action as a request to view the GT701's administrative interface. Instead, you must type in the local network IP address (usually 192.168.0.xxx) of computer B to access and use your Drupal site.

However, this causes a problem because `base_url` is used by Drupal to create all the site's links. If `base_url` is set for www.example.com, you will be unable to use the site from computer A (even if you type in the local IP). If `base_url` is set for the local network IP address, Internet users will not be able to use the site.

In other words, `base_url` has to be different for internal network users (computer A) and Internet users. To do this, replace the line `$base_url = "http://www.example.com";` with the following PHP code:

```
<?php
$ipaddress = getenv(REMOTE_ADDR);
$local= strpos($ipaddress, '192.168.');
if ( $local== false ) {
    $base_url = "http://www.yourdomain.com";
} else {
    $base_url = "local network IP address";
}
?>
```

Here, replace www.example.com with the actual URL and local network IP address with the proper IP address (something like 192.168.xxx.xxx). It is possible that your local network IP addresses do not begin with 192.168; in that case, you would need to change the code to look for the local IP range accordingly.

With this setup, you can access the Drupal site using computer A by typing in Computer B's local IP address in your browser, while Internet users can continue to access it by typing in www.example.com

Installing Drupal in a subdirectory

If you install Drupal in a subdirectory, you need to alter the .htaccess file in Drupal's root.

Change ErrorDocument to:

```
# Customized server error messages:
ErrorDocument 404 /subdirectory/index.php
```

Change RewriteBase to:

```
# Modify the RewriteBase if you are using Drupal in a subdirectory and the
# rewrite rules are not working properly:
RewriteBase /subdirectory
```

Remove any #'s in front of the RewriteBase line in case it's commented out.

Make sure your \$base_url in conf.php is set correctly as well.

More than one Drupal site on one machine

There are several possible configurations for running multiple Drupal servers on the same hardware. You can separate them by directories or by vhosts, they can share configurations or split them or, in some cases, have a mixture, but all of these methods have at their heart the ./sites/domain_or_host_name/settings.php configuration file and the search-sequence where the Drupal program will search first for a configuration named for the current page and then to the current host before settling for the default.

General rules for multiple Drupal deployments

Each of the possible multi-drupal scenarios is discussed in more detail in the sections that follow, but the general form for the alternate configuration filename is:

```
./sites/vhost.uri/settings.php
```

Note how the path separator ('/') must be changed to a dot. As an example, the vhost drupal.mysite.net may have one primary drupal server at the DOCUMENT_ROOT location, but a second site may begin at DOCUMENT_ROOT/altserver. For this case, the configuration file would be ./sites/drupal.mysite.net.altserver/settings.php

Within that configuration file, the most common and minimal option is to set the \$db_url that specifies the host, database and login for the Drupal tables, as well as the \$base_url. But you *can* also include assignments to override anything in the VARIABLES table. This allows you to redefine the theme, the site footer and contact email, blocks per-page limits, even the name you use for anonymous.

Drupal IDs: When using multiple drupal servers on the same hardware, each new configuration will result in a new *host* component for the `username@<i>host</i>` Drupal login ID (used when logging into a foreign Drupal server). For example, if you have a directory partitioned host at drupal.mysite.net/altserver your username to login to some other Drupal server would be USERNAME@drupal.mysite.net/altserver.

Multiple directories

Drupal allows you to setup multiple drupal sites using different directories on top of one physical source tree. This might be useful if you want to setup multiple sites about different topics (e.g. <http://yourdomain.com/travel/> and <http://yourdomain.com/sport/>) or if you want to provide users on your system with a personal drupal site (e.g. <http://yourdomain.com/~joe/> and <http://yourdomain.com/~john/>). When using Unix/Linux

as your host operating system, this can be best accomplished by using symbolic links:

```
$ ls -l includes/*.php
-rw-rw-r-- 1 drupal drupal includes/yourdomain.com.~joe.php
```

Once you created the configuration file, create a fake directory using symbolic links that matches the URI. For a drupal site with URI `http://yourdomain.com/~joe/` use:

```
$ ln -s . ~joe
```

If you want Joe to be able to configure his own drupal site, create another symbolic link to make the configuration file `includes/yourdomain.com.~joe.php` available to Joe in his home directory:

```
$ ln -s /path-to-drupal/includes/yourdomain.com.~joe.php /home/joe/
```

Multiple domains or vhosts

Multiple domains or vhosts using different databases

Apache supports both IP- and name-based virtual hosts (vhosts). While running more than one engine (by using vhosts) can be very useful for development and testing purpose, it might even be more interesting for hosting companies. Therefore, we tried to support vhosts in the best possible way in order to make the life of any administrator easier. We do so by making it possible to run an unlimited amount of vhosts on the same physical source tree, though by using different configuration files. Moreover, you can setup multiple configuration files in your `includes`-directory.

```
$ ls -l sites/*.php
-rw-rw-r-- 1 drupal drupal sites/www.yourdomain1.com/settings.php
-rw-rw-r-- 1 drupal drupal sites/www.yourdomain2.com/settings.php
```

The only thing left to be done is to setup the corresponding vhosts in your Apache configuration file. Note that the DocumentRoot points to the same source tree twice:

```
NameVirtualHost 127.0.0.1
DocumentRoot /home/www/drupal
ServerName www.yourdomain1.com
DocumentRoot /home/www/drupal
ServerName www.yourdomain2.com
```

Remember that as of Drupal 4.6, you can have site specific modules and themes as well. Just create a directory under the `sites/www.yourdomain1.com` called 'modules' and place the site specific modules. The same applies to themes as well.

Consult the `INSTALL.txt` that came with your Drupal installation for more details.

Multiple domains using the same database

If you want to host multiple domains (or subdomains) on top of the same database (e.g. `http://yourdomain.com/` and `http://www.yourdomain.com/`), simply use symbolic links to setup the required configuration files:

```
$ ln -s sites/yourdomain.com sites/www.yourdomain.com
$ ls -l sites
-rw-rw-r-- 1 drupal drupal      sites/yourdomain.com
lrwxrwxrwx 1 drupal drupal    sites/www.yourdomain.com -> sites/yourdomain.com
```

If your installation isn't in the root folder then you need to specify the path to the Drupal installation. For example if the URL to your installation is <http://www.example.com/drupal/> you would use `sites/www.example.com.drupal` with a `settings.php` file in it.

If you want cookies to be shared between two sites, you will need to set the value of PHP's `session.cookie_domain` correctly. In the case above, set it to ".`yourdomain.com`". You can do this through Drupal's `.htaccess` file.

Moving your Drupal installation to a new directory

If for instance you need to move your installation from `www.mysite.com/development/` to the root directory of `www.mysite.com`, just follow these simple steps:

Copy Files

Copy the files of your Drupal installation from the old directory to your new directory. Make sure you include `.htaccess`.

Change Path

In your new directory, open the file (4.5 or earlier) `includes/conf.php` or (4.6) `sites/default/settings.php`

Look for a line that begins with `$base_url =` , update this so that `$base_url` equals the path to your new directory. Save the file and close it.

You might need to modify the `.htaccess` file as well.

Update Cron

If you set up Cron on your old installation, make sure you update it to point to your new installation.

Delete Old Directory

Test that everything is working in your new installation. If so, it is now safe to delete the files in your old Drupal directory.

Linux specific guidelines

Installing PHP, MySQL and Apache under Linux

Installing MySQL shouldn't be too much of a burden, when using a Linux distribution that can handle RPMs. All you have to do is grab the RPMs from the MySQL website. Please do note that you'll also need the MySQL client RPM, not only the MySQL server one. Once MySQL has been installed, download Apache and PHP, and unpack them in the same directory. To install Apache together with PHP and MySQL, follow the "quick install"-instructions in the INSTALL-file located in your PHP directory. When configuring PHP do not forget to replace '*'apache_1.3.x'*' with your version of Apache.

After the compilation process you have to set the DocumentRoot in Apache's httpd.conf to the path of your drupal-directory. Make sure your Apache is setup to allow .htaccess files so drupal can override Apache options from within the drupal directories. Therefore, set AllowOverride to "All" instead of "None". Somewhat down httpd.conf they ask you to set Directory to whatever you set DocumentRoot to. The last thing to do is to add index.php in IfModule mod_dir.c behind DirectoryIndex. Apache will then look for index.php in the DocumentRoot and will display it as its main page.

Mac OS X-specific guidelines

Install and configure Mysql and PHP. Server Logistics provides nice pre-compiled packages and instructions. PHP is also available from Marc Liyanage.

The stock version of Apache should be fine.

Turn on "personal web sharing" in the sharing panel of System Preferences to start Apache.

In httpd.conf (in /etc/httpd), locate the following section and allow overrides, so that Drupal's clean urls will work (they depend upon rewrite rules in .htaccess). You'll need to be root (or sudo) to do this. Don't forget to restart apache after modifying httpd.conf (turn personal web sharing off, then back on again, or use /usr/sbin/apachectl restart).

```
#  
# This controls which options the .htaccess files in directories can  
# override. Can also be "All", or any combination of "Options", "FileInfo",  
# "AuthConfig", and "Limit"  
#  
#     AllowOverride None  
#     AllowOverride All
```

Drupal goes into /Library/WebServer/Documents/, or ~/Sites.

MacZealots has a tutorial on installing Drupal on Mac OS X 10.3.

Installing Drupal on Mac OS X 10.4 Tiger

Mac OS X 10.4 Tiger comes with PHP (though it's disabled by default) and without MySQL. Here's how to get Drupal up and running from a stock Tiger installation.

Installing and Configuring MySQL

Downloaded the latest MySQL installer from dev.mysql.com. I chose the Mac OS X Installer Package for 10.3.

I double-clicked the file that I downloaded (mysql-standard-4.1.11-apple-darwin7.8.0-powerpc.dmg), then double-clicked the mysql-standard-4.1.11-apple-darwin7.8.0-powerpc.pkg icon to run the installer and install MySQL.

Next, I installed the MySQLStartupItem.pkg because I want MySQL to start up when I start my Mac.

Then, I copied the MySQL.prefPane into the PreferencePanes folder in my Library folder at the root level of my hard drive. Tiger asked me to authenticate when I did this. Upon opening my System Preferences, I can indeed see a pretty MySQL icon.

I wanted to test to see if MySQL would really start up, so I restarted. Sure enough, MySQL was running.

Next, it was time to set initial passwords for MySQL. To do that, I opened the Terminal (in the Utilities folder). Then I typed in the following:

```
export PATH="$PATH:/usr/local/mysql/bin"
```

This tells your computer that when it interprets commands, it should also look in the freshly created MySQL installation for MySQL commands. If you want it added permanently instead of just for this terminal session, you could edit the third line of /etc/profile to read

```
PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/mysql/bin"
```

and then start a new Terminal window.

Next, we want to make MySQL secure by setting a MySQL root password. Fortunately a handy utility is provided to make that easy. I typed in:

```
mysql_secure_installation
```

Interacting with the script looked like this (note that at the first prompt I just pressed enter since no root password has been set yet):

```
Enter current password for root (enter for none):  
OK, successfully used password, moving on...  
Setting the root password ensures that nobody can log into the MySQL  
root user without the proper authorisation.  
Set root password? [Y/n] Y  
New password: zoinks  
Re-enter new password: zoinks  
Password updated successfully!  
Reloading privilege tables..  
... Success!  
By default, a MySQL installation has an anonymous user, allowing anyone  
to log into MySQL without having to have a user account created for
```

```

them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.
Remove anonymous users? [Y/n] Y
... Success!
Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n] Y
... Success!
By default, MySQL comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.
Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.
Reload privilege tables now? [Y/n] Y
... Success!
Cleaning up...
All done! If you've completed all of the above steps, your MySQL
installation should now be secure.
Thanks for using MySQL!

```

I verified that MySQL was running:

```

$ mysqladmin -u root -p status
Enter password: zoinks
Uptime: 938 Threads: 1 Questions: 16 Slow queries: 0 Opens: 21
Flush tables: 1 Open tables: 0 Queries per second avg: 0.017

```

It was running fine. Now on to the task of setting up MySQL for Drupal.

Creating the Drupal Database and Database User

Now that MySQL is installed and we have a root password established for it, it's time to create the database that Drupal will use:

```

$ mysqladmin -u root -p create drupal
Enter password: zoinks

```

The database was created. Time to create the user. The user in this example will be called *drupaldbuser* and the password for that user will be *fuffy*.

```

$ mysql -u root -p
Enter password: zoinks
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.1.11-standard
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> GRANT ALL PRIVILEGES ON drupal.* TO drupaldbuser@localhost IDENTIFIED BY 'fuffy';
Query OK, 0 rows affected (0.04 sec)
mysql> SET PASSWORD FOR 'drupaldbuser'@'localhost' = OLD_PASSWORD('fuffy');
Query OK, 0 rows affected (0.00 sec)

```

This last line was necessary because Tiger comes with PHP 4, and the MySQL libraries for PHP 4 use a different authentication protocol.

```
mysql> flush privileges;
Query OK, 0 rows affected (0.19 sec)
mysql> \q
Bye
```

Now I paused to rejoice. MySQL was installed and ready for Drupal. Now it was time to install Drupal itself.

Retrieving and Configuring Drupal

Development of the Drupal codebase is fairly quick, with a new release every six months or so. So I decided to use the CVS repository to download Drupal. That way I can use the "cvs update" command to keep my Drupal installation current with bug fixes and improvements.

I have the cvs option available to me because I installed the developer tools that were included on the OS 10.4 DVD. If you don't have them installed, or don't want to install them, you could always download Drupal directly and place the resulting directory inside

```
/Library/WebServer/Documents
```

First, I navigated to where I want my Drupal installation. Just to be consistent with Mac OS 10.4, I used /Library/WebServer/Documents:

```
cd /Library/WebServer/Documents
```

And now I pulled down the latest version of the Drupal 4.6 release:

```
cvs -z9 -d:pserver:anonymous:anonymous@cvs.drupal.org:/cvs/drupal checkout -r DRUPAL-4-6 drupal
```

This created a directory called drupal that contains the latest code for version 4.6. If, next week, I hear that a bugfix has been issued for 4.6, all I have to do is

```
cd /Library/WebServer/Documents
cvs update
```

Now it was time to create all the database tables that Drupal needs:

```
$ mysql -u drupaldbuser -p drupal < /Library/WebServer/Documents/drupal/database/database.mysql
Enter password: fuffy
```

Now I had MySQL running, and had a Drupal database populated by Drupal's fields. It was time to tell Drupal how to find that database.

In the Finder, I went to /Library/WebServer/Documents/Drupal/sites and duplicated the folder called default, and renamed it localhost. Then inside the new localhost directory I edited the settings.php file, changing line 81 to read:

```
$db_url = 'mysql://drupaldbuser:fuffy@localhost/drupal';
```

and changing line 90 to read:

```
$base_url = 'http://localhost/drupal';
```

The reason I duplicated the "default" settings folder was to avoid conflicts when I update CVS in the future, since CVS would have seen the changed settings.php file and tried to merge it with the original one. By keeping my settings file separate, I avoid this. As long as I access my Drupal installation through the URL http://localhost/drupal it will use the settings in the localhost folder.

PHP is not enabled by default on OS 10.4. So I enabled it by editing Apache's configuration file. I used BBEdit, but you can use any text editor you want. For example, you could choose Go To Folder from the Finder's Go menu and type in /etc/httpd, then double-click on httpd.conf and OS 10.4 will prompt you for an application to edit the file with. Here's me opening the file for editing in BBEdit:

```
sudo bbedit /etc/httpd/httpd.conf
```

Then I removed the # from in front of line 240:

```
#LoadModule php4_module libexec/httpd/libphp4.so
```

and line 284:

```
#AddModule mod_php4.c
```

and line 406:

```
AllowOverride All
```

After changing the configuration file, I stopped, then started the Personal Web Sharing service in the Sharing pane of System Preferences. This forces Apache to reread the configuration file and apply the changes. So now PHP was enabled.

Now, with my hands shaking from excitement, I entered http://localhost/drupal into my web browser. I was greeted with Drupal's welcome page:

Welcome to your new **Drupal**-powered website. This message will guide you through your first steps with Drupal, and will disappear once you have posted your first piece of content.

The first thing you will need to do is [create the first account](#). This account will have full administration rights and will allow you to configure your website. Once logged in, you can visit the [administration section](#) and [set up your site's configuration](#).

Drupal comes with various modules, each of which contains a specific piece of functionality. You should visit the [module list](#) and enable those modules which suit your website's needs.

Themes handle the presentation of your website. You can use one of the existing themes, modify them or create your own from scratch.

We suggest you look around the administration section and explore the various options Drupal offers you. For more information, you can refer to the [Drupal handbook online](#).

I clicked on the link entitled "create the first account", and saw:

user account

[log in](#) [register](#) [request new password](#)

Username:*
admin

Your full name or your preferred username; only letters, numbers and spaces are allowed.

E-mail address:*
[Placeholder]

A password and instructions will be sent to this e-mail address, so make sure it is accurate.

[Create new account](#)

Since this was the first account I created on Drupal, I knew it would have special privileges, so I named it admin and typed in my e-mail address. (Warning: naming your administrative account "admin" is bad security! Don't do this!) Then I clicked on the "Create new account" button. Drupal responded with a randomly generated password and an opportunity to log in immediately:

The screenshot shows the 'user account' page of a Drupal site. At the top, there are three buttons: 'log in' (highlighted in blue), 'register', and 'request new password'. A message below the buttons says, 'Welcome to Drupal. You are user #1, which gives you full and immediate access. All future registrants will receive their passwords via e-mail, so please configure your e-mail settings using the Administration pages.' Below this message, it says, 'Your password is **QX827Tbr6o**. You may change your password on the next page.' A note below that says, 'Please login below.' At the bottom left is a 'Log in' button.

I took the opportunity to log in immediately by clicking the "Log in" button. On the resulting screen I changed my password to something I could remember and clicked Submit.

Now I've got Drupal running on Mac OS 10.4. Time to celebrate with a swig of flavinoid-laden grape juice.

Windows-specific guidelines

Several packages exist which install Apache, PHP, and MySQL in one easy download. If you want to install them separately, see the guidelines below. Otherwise, have a look at

[Miniserver](#)

[Foxserv](#)

[PHPHome](#)

How to install Drupal for newbies using FTP and phpMyAdmin

There are three things to do: upload the database, get drupal ready and change "/tmp" on your drupal site.

Change "/tmp" on your drupal site.

1. Open a browser and go to your new drupal site, whatever its domain is.
2. Create a new user account. You enter a username and password and now you are the admin for your new site.
 - Important: the first user you create has access to every administration setting on your Drupal site.
3. On the left side of your screen, there is a link, "administer." Click "administer," then "settings."
4. Scroll down to "File system settings" and in the "Temporary directory" field you will see "/tmp." Remove the "/" from in front of "tmp." So now in "Temporary directory" the field should only contain "tmp." Click "Save configuration," at the bottom of the screen.

Get Drupal ready

1. Download Drupal
2. Extract Drupal to your desktop (or wherever you like on your computer). You need to have a program like 7-zip or Winzip installed first. You will have to extract the files twice.
3. Edit conf.php. It's in the drupal-4.5.2/includes directory after you extract the files. You have to edit two lines in this file.
4.
 - To edit the file, you may need to load your text editor (Windows comes with Notepad and Wordpad, either of which will do fine) first then find the file by clicking the File menu, then Open.
5. Change: \$db_url = "mysql://user:password@localhost/drupal";
 - You must replace "user" and "password" with your username and password for your phpMyAdmin login and "drupal" with your database's name.
6. Change: \$base_url = "http://www.example.com";
 - You must replace "example.com" with your domain name.
7. Put drupal on your server. You need an FTP program like FileZilla.
 - Open your FTP program and navigate to the directory where your index.htm file would normally go. This is usually called public_html/ or www/ This is where you put the drupal files. The drupal files that go there are the contents of drupal-4.5.2, that you extracted. You don't put the folder itself on the server, just the contents. That means you are uploading six folders (database, includes, etc.) and ten files (.htaccess, cron.php, etc.).
8. Create a new directory on your server.
 - In your FTP program, right click on the server side, select "create directory" and name the new directory "files." You want "files" in the main directory. After you create it, you should see it in between "database" and "includes."
9. Now you open up your new directory, "files," and create a new directory inside called, "tmp."
10. Change the file attributes of both "files" and "tmp" by right clicking on them one at a time, selecting "file attributes" and entering "755" in the "numeric value" field (if you see errors on your drupal site later saying "files" and "tmp" are not writeable you have to change the file attributes to "777," which is less secure).

Upload the database

1. Login to your phpMyAdmin (on your server).
2. Click "Create new database."
 - - name it "drupal" (or whatever you like).
3. You should see "drupal" at the top of the left column. Click "drupal."
4. In the center column you should see some tabs to choose from. Click "SQL."
5. At the bottom of the box that comes up it says, "Location of the textfile:" and there's a button that says, "Browse". Click "Browse" and navigate to the file "database.mysql." Its on your computer inside the folder you extracted, "drupal.4.5.2," inside the "database" folder. Select "database.mysql" and click "open."

Installing Apache (with PHP) on Windows

The first step to getting Drupal running on your Windows machine is to set up the Apache web server.

While you're at it, it's best to install PHP along the way because you'll be editing the same files for both of them.

1. Grab the latest copy of Apache and PHP.
2. Run the setup files and install the packages. It's best to do a full install.
3. When prompted for the directories to install the programs into, make sure there are no spaces in the paths. Oddly enough, the Apache-Installer defaults to **C:\Program Files\Apache Group\Apache**. If you keep this, you quite certainly will run into problems with cgi- and php-scripts not finding paths. Changing this to something like **C:\progs\web\Apache** and **C:\progs\web\PHP** will do just fine.
4. Go to the folder where you have installed Apache, and under that you will see a folder **conf**. In there is a file, **httpd.conf**, which you have to edit next:
 - Search for **ServerAdmin** and change it to your e-mail address
 - If you want to do local testing only, change **ServerName** to 127.0.0.1
 - Change **DirectoryIndex index.html** to **DirectoryIndex index.php index.html**
 - Change your **Documentroot** value to the folder where you unzipped Drupal.
 - Search for **AddType** and add the following lines:
 - **AddType application/x-httdp-php .php**
 - **AddType application/x-httdp-php-source .phps**
 - **ScriptAlias /php/ 'C:/php/'** where the path points to the folder you installed PHP to. Remember to use forward slashes.
 - **Action application/x-httdp-php '/php/php.exe'**
 - Find **<Directory** and change that value to **<Directory 'C:/Drupal'** with the same path as your **Documentroot** value.
 - Next, go to your PHP folder and edit **php.ini**. If there is no such file, check for a **php.ini-dist** and copy it.
 - Search for a section called **[mail function]** and fill in your outgoing mailserver (SMTP) and email-address.
 - Next, go to the section called **[Session]** and change the **session.save_path** to a valid temporary folder on your harddrive (e.g. **C:\Windows\Temp**)
 - Use the **Start Apache as a service** icon in your Start Menu. group.

Everything should work fine now.

Installing MySQL on Windows

1. After downloading the latest stable release version of MySQL, locate the **setup.exe**, and execute it. When prompted choose custom install.
2. Choose a path to install it. I recommend keeping to the default, which is **c:\mysql**.
3. Select all components, and continue until done.
4. Go to the MySQL folder, and start **winmysqladmin.exe**

5. Choose a username and password for yourself.
6. In the console, click on the my.ini tab, and choose "create shortcut in startmenu" option.
7. In the my.ini tab, also check that all the configurable options are correct in accordance to your computer.
8. Close the admin console and restart it. If admin program runs with a green light, everything fine.

Installing PHP4 on Windows

- After obtaining the latest stable release of PHP, extract the archive to `c:\php` or something similar.
- Copy `php.ini-optimized` to `php.ini`
 - Make the following modifications:
 - Change `include_path` to `"."`
 - Change `sendmail_from` to `"your@email.address"`
 - Change `SMTP` to `'your.smtp.mail.service'` If you don't know your smtp server, use the same configuration as your email client uses.
 - Change `session.save_path` to a temporary folder (`"drive:pathtotemp"`) and make sure the temporary folder exists.
 - Change `doc_root` to your preferred work folder (`"drive:pathtofiles"`). Make sure it exists and is the same folder you specified in the Apache setup procedure.
 - Set `register_globals` to equal "On"
- Save the changes, and copy the `php.ini` file to your Windows directory.
- Create a basic php file, for example `test.php` which contains the following:

```
<?PHP phpinfo(); ?>
```

and save it in your work folder.

- Open your web browser, and type in: <http://localhost/test.php>. If you get the PHP information page, then everything is set up correctly. If not, just go over the settings again for PHP to make sure everything is ok.

Installing PostgreSQL on Windows

Postgres is easily installed and administered on Windows. See PostgreSQL on Windows for the options you have.

- As of this writing, the apparently easiest choice for PostgreSQL on Windows is "UltraSQL by PeerDirect" mentioned at above link. It is available from here. See the README file enclosed in the download and Installing the PeerDirect PostgreSQL beta for Windows for more instructions. After completing installation, the username for your DB is your windows login name and there is no password.
- You might want to install phpPgSQL in order to admin your database. It will save you frustration at the command line. A more complete list of all known PostgreSQL GUI tools is available at PostgreSQL GUI's
- Go ahead and create your database tables via phpPgSQL or via the command line as described here.

Untar

When you download drupal packages you will need to decompress the files. Drupal packages are double compressed 'tar' and 'gz'. To untar packages in windows several programs are recommended.

- 7zip
- Winzip
- Winrar
- Winace

Using Clean URLs with IIS

Drupal can display brief, pretty URLs like those at drupal.org. For Apache sites, mod_rewrite powers this feature. For IIS, you will use a custom error handler for this. You probably want to disable logging in IIS, since every page view is considered an error using this technique.

- make sure your Drupal is working well without clean urls enabled.
- open your Internet Services Manager or MMC and browse to the root directory of the web site where you installed Drupal. You cannot just browse to a subdirectory if you happened to install to a subdirectory.
- right click and select properties -> custom errors tab
- set the HTTP Error 404 and 405 lines to MessageType=URL, URL=/index.php. If you are using Drupal in a subdirectory, prepend your subdir before /index.php
- paste the following code into the bottom of settings.php file, which is usually located under sites/default/. the first two lines should be edited. If you aren't using a subdirectory, set \$sub_directory to "". then set \$active=1 and enjoy!

```
<?php
// CONFIGURATION
$sub_dir = "/41/"; // enter a subdirectory, if any. otherwise,
use ""
$active = 0; // set to 1 if using clean URLs with IIS
// CODE
if ($active && strstr($_SERVER["QUERY_STRING"], ";")) {
  $qs = explode(";", $_SERVER["QUERY_STRING"]);
  $url = array_pop($qs);
  $parts = parse_url($url);
  unset($_GET, $_SERVER['QUERY_STRING']); // remove cruft added
by IIS
  if ($sub_dir) {
    $parts["path"] = substr($parts["path"], strlen($sub_dir));
  }
  $_GET["q"] = trim($parts["path"], "/");
  $_SERVER["REQUEST_URI"] = $parts["path"];
  if ($parts["query"]) {
```

```

$_SERVER["REQUEST_URI"] .= '?' . $parts["query"];
$_SERVER["QUERY_STRING"] = $parts["query"];
$_SERVER["ARGV"] = array($parts["query"]);
parse_str($parts['query'], $arr);
$_GET = array_merge($_GET, $arr);
$_REQUEST = array_merge($_REQUEST, $arr);
}
?
?>

```

- at this point, you should be able to request clean url pages and receive a proper page in response. for example, request the page /node/1 and hopefully you will see your first node shown. you should not use the q= syntax; use the clean url syntax. if you get an IIS error, you have a problem. please fix redo the above and then retest.
- browse to index.php?q=admin/system, enable clean URLs, and press Submit.
- you may get a php error if your php error reporting in your php.ini file is set to high. Try this setting in your php.ini file

error_reporting = E_ALL & ~E_NOTICE

Windows XP IIS development/test system guidelines

Note, this guide is for setting up a development/test site on a Windows XP system. It is NOT suitable for use in setting up a Windows IIS server on the Internet. It does not take into account basic steps in securing the underlying OS or an IIS server against outside hacking. Windows XP is NOT Suitable for site hosting on the Internet but is nice for a locally hosted development system.

Assumptions of this document.

system name: COMP1

database name: site_drupal

database user: site_user

password: changeme

Choose a naming convention of some sort. While your intention may only be to have one site, do not rely on this. Even when testing, practice good standards so that you may establish a habit of good practices.

all downloads are downloaded to a c:\support directory

applications are installed in a c:\app\ directory

Web site is localhost and installed in the default location of c:\inetpub\wwwroot

You are familiar with how to use Google to search for answers and perform troubleshooting in the appropriate forums.

NOTE: It is a VERY BAD security practise to install a production IIS website with InetPub on the drive with OS (usually the C: Drive).

=====

Download the following in preparation.

Download php to c:\support\php

<http://www.php.net/downloads.php>

This example uses PHP 4.3.6 installer

Download the MySQL installer to c:\support\mysql

<http://dev.mysql.com/downloads/mysql/4.0.html>

This example uses 4.0.18

Download MySQL Control Center to c:\support\mysql

<http://dev.mysql.com/downloads/mysqlcc.html>

This example uses 0.9.4

Download Drupal 4.4.2 to C:\support\drupal\core

Download Modules of interest to c:\support\drupal\modules

=====

Windows XP

Go to add/remove programs.

Windows components

highlight Internet Information Services

-select details

-select World Wide Web service

Click ok or finish

Close out Control Panel

Browse to <http://localhost>

You should get the Welcome to Windows XP Server Internet Services page

-if not, troubleshoot this problem before you continue.

=====

Now to the installs

Run the php installer

-when it asks for a directory location, install to C:\app\PHP

-Accept all the defaults (allow IIS 4.0 compatible)

right click C:\app\PHP\sessiondata and select Sharing and Security

-choose the security tab

-Click add - advanced -find

-select IUSR_COMP1

-click the write Allow box and ok

Do the same for C:\app\PHP\uploadtemp directory

Right click php.exe (and php4ts.dll) select Properties

-click the Security tab

Add / advanced / Find now

-select IUSR_COMP1

-The Read & Execute box is selected by default.

php is now installed

From your Administrative tools menu, Launch Internet Information Services Admin

Go down to your Default Web Site

-right click and select properties

-select the documents tab and click add and add index.php

-click ok and out

Open up Explorer and browse to C:\Inetpub\wwwroot and delete everything in it.

=====

Note: There are probably better ways to do this but this consistently works for me.

Install MySQL -change the install directory to c:\app\mysql

Accept all the other defaults.

browse to C:\app\mysql\bin and launch winmysqladm.exe

log in as user root and leave the password blank. (we're only using it to create the my.ini file)

anyway.

choose the my.ini tab,

uncomment port=3306

choose save modifications

close and relaunch winmysqladm

MySQL should now be running and responding on port 3306, you can test by telnet localhost 3306.

If not, troubleshoot before proceeding.

Now, let's set a root password.

c:\app\mysql\bin> mysql -u root

```
mysql> UPDATE mysql.user SET Password=PASSWORD('changeme') WHERE User='root';
```

```
mysql> FLUSH PRIVILEGES;
```

```
mysql> /q;
```

=====

Install MySQLCC accept all the defaults.

Launch MtSQLCC

Name: local

Host Name: localhost

User Name: root

Password: changeme

Select Add.

right click on local and choose connect

right click on Database and choose Create new database.

```
enter site_drupal
right click on site_drupal and connect
right click on the Users table -create new user
Username: site_user
Host: localhost
Password: changeme
Check site_drupal
Check All Privileges mysql -u site_user -p site_drupal < database.mysql
Enter password: changeme
```

If you get a c: prompt, then there were no errors. If there were errors troubleshoot and solve before you move on. This is not a Drupal issue, this is a mySQL issue. Choose the support forum accordingly.

You can view your success by using MySQLcc.

Open C:\Inetpub\wwwroot\includes\conf.php in an editor.
(I prefer Crimson Editor www.crimsoneditor.com as it has context highlighting for php among others but choose your favorite)
Change line 17 per the Install.txt

```
$db_url = "mysql://site_user:changeme@localhost/site_drupal";
Line 31 will work as is, but it is a bad habit to ignore it, so change it to your computer name
$base_url = "http://comp1";
```

Alternatively you can replace it with this line that I found in the forums. There may be consequences to it's use that I am unaware of.

```
$base_url = "http://". $_SERVER['HTTP_HOST'];
(If you use host headers it will adapt nicely to all of them).
```

You are now at step 5 of Drupal's Install.txt.

You can use the Windows Scheduler to run cron.php. I do mine every four hour. Yours will depend on your site config. Open scheduler in the control panel. Add scheduled task. Name it something and choose daily. Click next and schedule a start time Every Day. Choose an account to run it under (you can set up a service account for this) Check the advanced properties box and finish. Select the Schedule tab, then advanced. Schedule accordingly. Under task, edit run to read, C:\PROGRA~1\INTERN~1\iexplore.exe "www.sample.com/cron.php". Click ok. Done.

Installing Drupal on Windows

[this page used to contain verbose windows installation guidelines. they got removed because they were a) just a copy of the general installation guidelines, b) misleading ("start by extracting the archive to the PHP working folder"), and c) we don't want to maintain redundant documentation. this page should only contain windows specific guidelines that *differ significantly from the general guidelines*. preferably, latter would be put so generally that we don't need anything here (eg. don't rely on "wget" and "tar" etc.)]

Installing Drupal on Windows Ext

Table of Contents

- I. Requirements
 - II. Installation
 - III. Connecting it All Together
 - IV. Drupal Adminstration
 - V. Setting Permissions
 - VI. Customizing Themes
 - VII. Scheduling Tasks
 - VIII. Optional Components
 - IX. Upgrading an Existing Drupal Site
 - X. More Information
-

Requirements

Drupal requires a web server, PHP4 (<http://www.php.net/>) and MySQL or a database server supported by the PHP PEAR API (<http://pear.php.net/>).

NOTE: The Apache web server and MySQL database are strongly recommended; other web server and database combinations such as IIS and PostgreSQL are possible but tested to a lesser extend.

I strongly recommend a complete web server packaged installer if you are at all new to Apache, MySQL or PHP. There are many out there. My personal favorite (because it worked out of the box) is FoxServ (<http://sourceforge.net/projects/foxserv/>).

Server Configuration

Your PHP setup must have the following settings enabled. These can be set in the php.ini file:

session.save_handler user

In addition, we recommend the following settings:

session.cache_limiter none

The php.ini file is usually found in the WinNT directory. These settings can be set in .htaccess file (in the drupal directory) overriding whatever is set in the php.ini file. There is a very helpful function in PHP that gives you all the information about how PHP is setup on your server. You can find this information out easily with PHP's `phpinfo()` function. This function also shows you where your php.ini file is located so you can make any changes there. To find out about your PHP settings simply create a file called `phpinfo.php`. Enter one line of text, "
`<?php`

```
echo phpinfo();  
?>
```

" and save the file to your server (where php is installed). For example, copy it to your "www.mydomain.com" or "localhost" directory and view it in a browser. Be sure to have apache running when you test it.

Installation

Step 1.

Downloading Drupal

You can obtain the latest Drupal release from <http://drupal.org/>. Click the downloads link. Download the most current tar.gz format and extract the files. You may need a tool to uncompress the files. I recommend picozip (<http://www.picozip.com/>) because it is easy and supports a huge number of compression formats. At the time of this writing it has a 30 day trial period.

This will create a new directory drupal-x.x.x/ containing all Drupal files and directories. This directory can be several directorys deeper than the unzipped directory. The directory we are concerned about has the index.php page and modules directory in it. Move the contents of that directory into a directory within your web server's document root or your public HTML directory. On my local machine I created a directory of the name of the domain name it is part of. For example, on my local machine I copy the files to, "C:\FoxServ\www\drupal". I would recommend copying it to a directory but if you are ready you can copy the files to root of the site which would be something like, "D:\FoxServ\www\" (locally) or "ftp://www.mydomain.com/www" or "ftp://www.mydomain.com/var/www/html".

NOTE: when copying files, ensure you also copy the hidden .htaccess file. You can see hidden files in Explorer by going to the menu item Tools > Folder Options > View > Hidden Files and Folders > Show Hidden Files.

Step 2.

Creating the Drupal Database

These instructions are geared toward a MySQL database. If you are using another database and you know a little bit about databases you should be able to follow along quite nicely. Be sure to check the database documentation for your specific database if you have any questions.

For this part of the tutorial I am going to use MySQL-Front to create and setup our Drupal database. At the time of this writing it is free. You can goto and download MySQL-Front from <http://www.mysqlfront.de/>. Tell the guy thank you and donate. I received an error that prevented the program to launch when I tried to "Launch Program Now" from the installation program but on a second attempt the Start > Programs > menu it launched successfully. Alternatively you can use MySQL.exe from the command line to achieve the same thing. I will list the command line instructions after the MySQL-Front instructions.

To follow alongin the next steps you will need to login to your MySQL database with a user account that has the CREATE and GRANT privileges. You will need to use the appropriate user name for your system.

Creating the Database with MySQL-Front

If you are going to create a database from the command line skip to the next section, otherwise continue on here.

First, you must create a new database for your Drupal site. To do so:

- Opening MySQL Front
- Find and open MySQL-Front. An Open Session window will appear.
- Creating a New Connection Session
- Click the New button to create a new connection session for drupal. An Add Session window will appear.
- Under the Common tab enter the name for your connection. In this tutorial we will call it "Drupal Connection".
- Switch to the Connection tab and enter the name of the server that has mysql running on it. This would be "www.myserver.com" or an IP "127.0.0.1". Enter "localhost" if you are running it locally. The default port for mysql is "3306". Default Timeout is 30 seconds. Connection Saver is active by default. I dont know what Connection Saver is.
- Switch to the login tab and enter the username that you setup when you installed mysql. You have the option to choose the database to startup in. You will choose this at another time because we have not created our database yet. Click the Ok button to save your new mysql session connection and to take you back to the Open Session window. If you dont know your're userid and password you will need to contact your hosting company or whoever set it mysql on your server and get a userid or send them these instructions. If you installed mysql yourself the username is "admin" or "root" and the password is blank (on a fresh mysql installation).
- Login to MySQL
- Select the new session you just made and click the OK button. A window will prompt you for your password if a password was not supplied (when you were creating the session connection). If your password is blank you do not need to enter anything here. Otherwise enter your password now. Click OK to login to your mysql account.

If you get any errors the program will let you know about it and sometimes offers accurate advice on what to do to fix it. Try what it says and if it doesn't work use the Command line method listed below or contact the authors of MySQL-Front at <http://www.mysqlfront.de/>.

Assuming you login successfully you will be shown a list of databases attached to the mysql server.

- Creating a New Database
- To create our new drupal database, select Database > New > Database from the menu bar. There is also a toolbar icon that adds a new database. For this tutorial we will try to stay with

menu commands.

- A New Database dialog appears.
- Enter the name of your new drupal database. If you are running it on your local machine I would name it, "drupal" or the same name as your domain, "mydomainname". I recommend using all lowercase in the names you specify to avoid case sensitive errors later on. MySQL will then create the initial database files.
- Creating a User with Access Rights
- Next you must set the access database rights. Right click on Users in the Host tree and select Database > New > User from the menu bar. An Add User window will appear.
- In the Common tab enter the name of the new user. In this tutorial we will use, "dba_user". In the password field enter a password. If you are smart you will write these down now. If you are not then skip it.
- Switch to the Hosts tab and enter "localhost" or the name of your session connection. You will get an error in german if you enter the wrong hostname. Interpreted it says, "I am german. No speeka englace." Ok, I dont seem to be doing something right here. Let's skip this method. Cancel out of that. We are going to run a script in MySQL-Front's SQL Query.
- Click on the SQL Editor and copy this code into it:

```
GRANT ALL PRIVILEGES ON myDatabaseName.* TO myUserId@myDomainName  
IDENTIFIED BY 'myPassword';
```

- Substitute "myDatabaseName" with the name of your database. Substitute, "myUserId" with the name of the drupal account that will be responsible for administering your drupal database. Substitute "myDomainName" with the name of of your domain. If you are running drupal on a local machine enter, "localhost". Finally substitute, "myPassword" with the password for your "myUserId@myDomainName" user ID. If you are smart you will write your username and password down now.
- Select Database > Run from the menu bar. If successful you will get "0 rows affected" in the MySQL-Front status bar. This will add a new user with permissions necessary to administer your drupal database.
- Let's verify that we created a new user. Click on Users in the Host tree and select View > Refresh from the menu bar. If everything went hunky dory then we should see our new user listed. If not then check the error messages, go back and check for misspelling and incorrect syntax and try again.

- Importing the Drupal Database Scheme

Once you have created your new database, you must load the required drupal tables into it. To do this you must go to the "Importing the Drupal Database Scheme" section in the Command Line section or obtain an additional file called, "database.sql". This is a MySQL import compatible MySQL database. MySQL-Front cannot import "*.mysql" files which are the only kind included with drupal distribution. At the time of this writing the "database.sql" is not included with the drupal distribution. I made the file because I imported the database from the command line and once it was in MySQL-Front I exported it to "database.sql". It took 10 seconds

to export it to ".sql" file. And ten seconds to import it to my test database. But whatever, you can write me or request it from the HEAD team at drupal.org. It would be better to have them manage it.

- Right click on your new drupal database in the Host tree and select Import > SQL File. Browse to the "[drupal install dir]/database/database.sql" directory and select the file, "database.sql". This is the SQL script that will create your drupal tables. An Import Options window will appear. Click Ok and your database will be created.

If you did not receive any errors then you have just created your drupal database! Yea!

Creating the Database from the Command Line

If you have already created a database using MySQL-Front skip to the next section. Otherwise you aren't going anywhere buddy. You gotta a lesson in "Running MySQL from the Command Line".

- Creating the Database

- You must create a new database for your Drupal site for it to work. The tool to do this is the mysql.exe file that can be run from the command line. To get to the command line goto Start > Run and enter "cmd". Click the OK button. You will be presented with a scary black window with a square flashy thing. This is called the command line. (cue ghost sounds). This is what people used to work in before graphical user interfaces available.

- Browse to the directory where "mysql.exe" resides. The two commands you use to browse are, wait for it, "dir" and "cd". "Dir" lists the contents of the directory and "cd" changes the directory. Ok I was going to explain this but if you are installing a web site then you dont need to know this. (...ok fine.) Change directories to the mysql directory. Enter "cd\" to get to the root of your hard drive and then enter "cd FoxServ\mysql\bin". Well, on my machine the directory to the "mysql\bin" folder is "D:\FoxServ\mysql\bin". If you did everything right command prompt will look like this, "d:\FoxServ\mysql\bin>". If you see something like that then pat yourself on the back. You are a big boy now.

- Now enter the following where "dba_user" is the name of the user id that has database administration rights to create your drupal database and "drupal" is the name of the drupal database to create:

```
mysqladmin -u dba_user -p create drupal
```

- SQL will prompt you for the dba_user database password. Enter the password and hit enter. Note: if you just setup your mysql to run on your computer or server or whatever then you DO NOT HAVE A PASSWORD YET!!!! AAHHHHAHGGGG! That is ok. Calm down. You can set it later. For now just leave the password field blank and press the entertainme key. If you receive no messages and are back to the command prompt then you have just created the your drupal database. We still need to import the database tables and setup a user before we can go further.

- Setting Access Rights

- Next you must login into your new database and set the user's database access rights. To do that we need to log into the MySQL command prompt. This is a command prompt that

mysql creates inside the bigger command prompt. What? You closed the command line window? No, no NO! This wont work at all! That's it I quit. (What drupal slave master? I am bound by the GPL to finish writing this on pain of death? Is it death by snoose snoose? No? Hmm. Ok. Fine. I'll keep going.) Ehem, excuse me, where was I? Oh right, let's get that command prompt back up.

- Make sure we are in the same directory as before and enter the following where "dba_user" is the name of the user id that has database administration rights:

```
mysql -u dba_user -p
```

- Again, you will be asked for the dba_user database password. Enter it (or dont if you dont have one) and press enter. You will now be at the mysql command line prompt which looks like "mysql> ". pretty isn't it?

- Creating a new user with permissions

- Now we need to create a new user that drupal can use to have access rights to the database.

- At the MySQL prompt, enter following command where 'myDrupalDatabase' is the name of your new drupal database, "myDrupalUserID@localhost" is the new Drupal UserId, "localhost" is the server where MySQL is installed and running and "myDrupalUsersPassword" is the new password for "myDrupalUserID@localhost" required to log in as the MySQL user. You must include the semicolon at the end of the line for MySQL to evaluate your statement. Enter the following and press Enter:

```
GRANT ALL PRIVILEGES ON myDrupalDatabase.* TO myDrupalUserID@localhost  
IDENTIFIED BY 'myDrupalUsersPassword';
```

- If this attempt is successful, MySQL will reply with "Query OK, 0 rows affected < 0.08 sec >". Note: You must remember to include the semicolon at the end of every statement you enter at the mysql prompt, otherwise it just sits there, waiting, waiting, waiting...

- Activating Permissions

- You must activate the new permissions for MySQL to apply the last step to the current running databases. To activate the new permissions you must enter the the following from the mysql command prompt:

```
flush privileges;
```

- and press enter. This refreshes and applies the permission to the new user we just made.

- Exit MySQL Command Prompt

- We must now exit the mysql command prompt to finish creating our database. To exit mysql command prompt type 'exit' and press enter. You will be returned back to the command prompt.

- Importing the Drupal Database Scheme

- Once you have created your new database, you must load the required drupal tables into it. To do so enter the following from the command prompt:

```
mysql -u myDrupalUserID -p myDrupalDatabase < database/database.mysql
```

- where "myDrupalDatabase" is the name of your new drupal database, "myDrupalUserID" is the new MySQL userid you just created(without the "@localhost") for use with your new drupal database and "database/database.mysql" is the path to the mysql database stored in the "[drupal install dir]/database/" directory. Now would be a good time to copy the database (using windows explorer) to the same directory where "mysql.exe" resides (that is the directory you are in right now). On my machine it is "D:\FoxServ\mysql\bin". When entering the userid remember to leave off the "@domainname" we specified in previous steps. It's was only necessary when we created a user but not used after that.

- You will be prompted to enter a password. Enter the password you created for your "myDrupalUserID" account. If you receive absolutely no messages whatsoever then the drupal database was successfully imported. That is called living by faith. Believing it worked even when you see no sign. If you do receive an error then I suggest checking the specified the path to the database and spelling errors.

- Close the Command Prompt window

- Type "exit" and press enter to exit the command window.

If you've got this far then you can conclude that your drupal database was installed and setup successfully.

Connecting it All Together

We are almost done. Before Drupal will work we need to setup a few more settings. Drupal server options are specified in includes/conf.php file. Before you can run Drupal, you must set the database URL and the base URL to the web site. The database URL creates a connection string to connect to your database. Remember how I said you can have multiple drupal sites running? Here is where you can configure that.

Setting the Path to the Drupal Database

- Browse to the "includes" directory in your drupal server install directory and open the "conf.php" file in a text editor. On my machine the "conf.php" file is located in "D:\FoxServ\www\includes\conf.php". On my second drupal install, my test site, the "conf.php" file is located in "D:\FoxServ\www\mytestsuite\includes\conf.php". I have two duplicate sites on my server. I've copied the orginal files to both the root "D:\FoxServ\www" and the test directory, "D:\FoxServ\www\test". Open the "conf.php" configuration file (in whatever drupal install you are in) and edit the \$db_url line to match the values we defined in the previous steps:

```
$db_url = "mysql://myDrupalUserID:myPassword@localhost/myDrupalDatabase";
```

- This step is essential. This is the login script to get into your specific MySQL drupal database.

Setting the Path to the Drupal Directory on your Server

- This step is also essential. If you dont get this you will only see the home page. But nothing else. The base URL is the location, the directory where your index.php file exists on your server. So if you put your drupal files into a directory say, "http://www.mysite.com/drupal" then you would put the same thing here.

- Set \$base_url to match the address and directory of drupal on your web site:

```
$base_url = "http://www.mysite.com";
```

Here are some more examples. Use only one that matches your specific configuration:

```
$base_url = "http://www.mysite.com/directoryWhereDrupalIs";  
$base_url = "http://localhost";  
$base_url = "http://localhost/anotherDrupalSite";
```

- NOTE: for more information about multiple virtual hosts or the configuration settings, consult the Drupal handbook at drupal.org.

Setting up more than one drupal site on one machine using Virtual Hosts

Besides the method I've mentioned above Drupal also allows for multiple virtual host installations. To configure a virtual server host, make a copy of the conf.php file we worked with earlier and rename it to "www.yourdomainname.com.php". place this in your includes directory. NOTE: This part of the instructions (setting up virtual hosts) is not tested. If someone validates this and would include any steps I have forgot then i will add it here for the benefit of others.

Drupal Adminstration

You can now launch your browser and test your site! Browse to the root directory where Drupal is installed. Make sure Apache and MySQL services are running. If everything is setup correctly you will be at the home page of your new Drupal site. Upon a new installation, your Drupal website defaults to a very basic configuration with only a few active modules, one theme, and no user access rights.

Use your administration panel to enable and configure services. For example, set some general settings for your site with "Administration - configuration". Enable modules via "Administration - configuration - modules". User permissions can be set with "Administration - accounts - permissions".

For more information on configuration options, read through the instructions which accompany the different configuration settings and consult the various help pages available in the administration panel.

Note that additional community-contributed modules and themes are available at <http://drupal.org/>.

Setting Permissions

Out of the box you will not have permission to do or view anything but the home page. Neither will anyone who visits the site. You have to give them (an anonymous visitor) privileges to view your site.

The first thing you need to do is create a master user account. Follow the on screen instructions to create an account and login. The first account will automatically become the main administrator account for the site. You can add additional administrator accounts at any time. I recommend creating another account with administrative permissions for security reasons and using that from now on. Write your user name and password to both down in a secure location.

Creating the Administrator Role

By default Drupal only has two roles, anonymous visitor and authenticated user. Except in certain cases you will want to create additional roles for the different users that use your site. For now go to Administer > Accounts > Roles and add "Administrator". Enter "administrator" and press the Add button.

Giving Permissions to a Role

By default all the roles have very limited capabilities. New roles have no permissions set. If we are going to create a new administrator user account for ourselves we will need to change this. We can change this in the Permissions page. Goto Administer > Accounts > Permissions. Here is a table filled with options for all your different roles. Set your administrator account to have all permissions for now and click save permissions. You can go back and change this after you get familiar with Drupal (think a few months from now). Please note. This is the page to give users the permission to see the content on your site. To enable anonymous users to see your site content find a row called, "access content" check the checkbox.

Creating a New User

Finally, we are one step away from creating our secondary administor account. Goto Administer > Accounts > New User. On this page enter the new account information and click Create Account. You can now log out of the administrator account (write down the password) and login with your secondary administrator account.

Customizing Themes

Now that your server is running, you will want to customize the look of your site. Several sample themes are included in the Drupal installation and more can be downloaded from drupal.org.

Customizing each theme depends on the theme. In general, each theme contains a PHP file themename.theme which defines a function header() that can be changed to reference your own logos.

Most themes also contain stylesheets or PHP configuration files to tune the colors and layouts; check the themes/ directory for README files describing each alternate theme.

Scheduling Tasks

Many Drupal modules have periodic tasks that must be triggered at specific intervals. This is called a cron job and they are setup in the cron.php page. To activate these tasks, you must call the cron page; this will pass control to the modules and the modules will decide if and what they must do.

The following example crontab line will activate the cron script on the hour:

```
0 * * * * wget -O - -q http://HOSTNAME/cron.php
```

More information about the cron scripts are available in the admin help pages and in the Drupal handbook at drupal.org. Example scripts can be found in the scripts/ directory. I dont know the Windows equivalent of this section. Please email me if you know.

Optional Components

- To use XML-based services such as the Blogger API, Jabber, RSS syndication, you will need PHP's XML extension. This extension is enabled by default in standard PHP4 installations.
- If you want support for clean URLs, you'll need mod_rewrite and the ability to use local .htaccess files. (More information can be found in the Drupal handbook on drupal.org.)

Upgrading an Existing Drupal Site

1. Backup your database and Drupal directory - especially your configuration file (www.example.com.conf or includes/conf.php).
2. Log on as the user with user ID 1.
3. Overwrite all the old Drupal files with the new Drupal files.
4. Modify the new configuration file to make sure it has the correct information.
5. Run update.php by visiting <http://www.example.com/update.php>.

More Information

For platform specific configuration issues and other installation and administration assistance, please consult the Drupal handbook at <http://drupal.org/>. You can also find support at the Drupal support forum or through the Drupal mailing lists.

MS SQL Server Guidelines

Update: with Drupal 4.4, MSSQL is not supported because we have no maintainer for this piece of the application. If you wish to update the database.mssql schema and get MSSQL working again, please send a note to the drupal-devel mail list.

In order to use MS SQL Server, you will need the following:

- PHP with the MSSQL extension active
- PEAR must be installed and on your include path. You can set the include path in your conf.php with something similar to ...

```
ini_set("include_path", ".;c:/php/pear");
```

Add the following line to your includes/conf.php file:

```
ini_set("magic_quotes_sybase", 1);
```

Use Query Analyzer or Enterprise Manager to do the following:

- Create a database for your site.
- Create a user who has may read/write data, and create/delete tables.
- Once you have a proper database, dump the required tables into your database by executing the file database.mssql in your Query Analyzer.
- Note that the bottom of the database.mssql file contains function(s) which only work in SQL 2000. If you are using a prior version, you currently cannot use the forum and tracker modules. These functions seem not to work without minor modification to the Drupal source code. Specifically, substitute dbo.GREATEST wherever you find GREATEST. Please post here if you find a way around this.
- Set the database options in includes/conf.php so Drupal can access the database you have created. Edit the following line in includes/conf.php:

```
$db_url = "mssql://username:password@hostname/dbname";
```

PostgreSQL specific guidelines

1. Create a PostgreSQL database for your site.

```
createdb -U username dbname
```

where *username* is the owner of the database (this user must have permission to create databases) and *dbname* is the name of your database. You will be prompted for that user's password. On success, the following is displayed: CREATE DATABASE

2. Once you have a proper database, dump the required tables into your database:

```
psql -u username dbname < database/database.psql
```

You will be prompted for your database password. You should see a progress report as the tables are created. All has gone well if there are no lines marked "Error:" printed to the screen.

3. Set the database options in includes/conf.php so Drupal can access the database you have created. Edit the following line in includes/conf.php:

```
$db_url = "pgsql://username:password@hostname/dbname";
```

Installing new modules

After installing Drupal, you have the option of installing extra modules to extend or alter Drupal's behavior.

In brief. Download the module, extract, upload the folder into your Drupal modules folder, run the mysql file if necessary, and enable the module in *administer* → *modules*.

1. Download the new module. Make sure the version of the module is compatible with your version of Drupal. Also note that modules labeled CVS are considered unstable and should be handled with care. Typically, CVS modules work only with the CVS version of Drupal.
2. Extract the module. When you first get the module, it will probably come in a compressed file format such as tar.gz. On Windows, use a program like WinZip to extract it. On the Mac, you can use Stuffit Expander. To extract the file using the Unix command line:

```
tar -zxvf modulename-4.5.tar.gz
```

You should see a list of files extracted in to a folder.

3. When you've extracted the file, upload the files via FTP to a folder inside the modules/ folder of your Drupal installation.
4. Read the installation file (usually INSTALL.txt and/or README.txt). Sometimes the installation file has no extension, so when you try to click on it, your computer doesn't know what program to use. In that case, open Notepad (or your favorite text editor) first, and then open the file into it.
5. There are modules that modify the database. You can generally tell if there is a

`modulename.mysql` file included with the module. If you do have to modify the database, see the next few steps. If you do not, please skip to step 7.

6. If you have to modify the database to get your module running, you will need to add tables to the database you made when you installed Drupal.
 - **Using phpMyAdmin:**
 - If you have phpmyadmin, log in and go to your drupal database. If you have it, but do not know how to access it, please contact your host.
 - Click on the tab that says 'SQL'
 - You should see a text area labeled 'Run SQL query/queries on database `yourdrupaldatabase`'. Underneath, it says 'Or Location of the textfile:' Click browse, and find the `modulename.mysql` that came with the module. Click go. Unless your instructions for the module says anything else, that should be all you have to do to the database.
 - **Using the Unix command line:**
 - Run the following command.
 - `mysql -u username -ppassword database_name < modulename.mysql`
 - Replace 'username' with your MySQL username, 'password' (but keep the -p before it) with your MySQL password, 'database_name' with the database Drupal uses, and 'modulename.mysql' with the SQL file that the module comes with. You can generally find this out from `settings.php` in either the `sites/default` folder or the `sites/sitedomain.com` folder, replacing `sitedomain.com` for the domain that hosts Drupal.)
7. For most modules, all that is left is to activate it! To activate your module, you need to click *administer » modules*, check the box next to your new module name, and click on 'Save configuration' at the end.
8. Some modules will require you change permissions or settings to get them working as you like them. Permissions and settings info may be in the instructions that came with the module. If not:
 1. Click *administer » access control*. Scroll down to see if the module appears in the list and, if it does, give the appropriate permissions to roles.
 2. Click *administer » settings* and see if the name of the module you just installed is in the list. If it is, click the module name and configure as appropriate.
 9. If you still run into the problems, search the forums. If your problem hasn't already been addressed, post a new post.

Installing new themes

Once you get Drupal installed and you start to come to terms with it you will probably want to customize the way it looks.

There are several themes which you can download from the Drupal web site which should get you started.

Installing a new theme is very straightforward:

1. Download a new theme package. Note that themes for different Drupal versions are not compatible, version 4.4 themes do not work with Drupal 4.5 and reverse.
2. Read any README or INSTALL files in the package to find out if there are any special steps needed for this theme.
3. Check to see if you have the required theme engine to be able to display your theme. Theme engine files go in a folder in themes/engines in your Drupal directory.
4. Upload the contents of the theme package to a new directory in the themes directory in your Drupal site. For example, themes/box_grey.
5. Click *administer » themes* and enable the new theme (Drupal will auto-detect its presence).
6. Edit your user preferences and select the new theme. If you want it to be the default theme for all users, check the *default* box in the themes administration page.

Tuning your server for optimal Drupal performance

There is quite a lot of tuning that can be done to your web server and its supporting software to increase the ultimate performance of Drupal. This document is an attempt to compile into one place the many tuning tips that have proven beneficial to other Drupal users.

If you have something you can add, by all means please do!

Tuning PHP

1. If you have CPU cycles to spare, or if bandwidth is a more constrained resource than CPU cycles, you can add the following to php.ini:

```
output_handler = ob_gzhandler
```

A comment in php.ini explains:

"You can redirect all of the output of your scripts to a function. For example, if you set output_handler to 'ob_gzhandler', output will be transparently compressed for browsers that support gzip or deflate encoding. Setting an output handler automatically turns on output buffering."

This functionality is further described here.

2. very interesting presentation from a PHP developer

Additional resources:

- PHP Project Page

PHP caches

PHP is a scripting language. Each time a PHP script is run to generate a webpage with Drupal, your web server must compile the PHP script into an executable format. This results in an obvious amount of overhead each time a page is generated.

A PHP cache can be installed to save and re-use compiled PHP scripts, thus greatly reducing the amount of overhead required for Drupal to display a web page.

There are a number of PHP caches (aka accelerators) available, including:

- Turck MMCache
- PHP Accelerator
- Alternative PHP Cache (APC)
- After Burner
- Zend Accelerator
- eAccelerator

Turck MMCache

The Turck MMCache has been confirmed to work well with Drupal. Installation is quite simple, resulting in a quick and noticeable performance increase.

Overview:

According to the project's home page:

"Turck MMCache is a free open source PHP accelerator, optimizer, encoder and dynamic content cache for PHP. It increases performance of PHP scripts by caching them in compiled state, so that the overhead of compiling is almost completely eliminated. Also it uses some optimizations to speed up execution of PHP scripts. Turck MMCache typically reduces server load and increases the speed of your PHP code by 1-10 times."

Compatibility:

The Turck MMCache runs on Linux and Windows, working with Apache 1.3 and Apache 2.0, compatible with PHP 4.1 and later.

The following versions of MMCache have been tested successfully with Drupal 4.1+: 2.3.15, 2.3.23, 2.4.6.

Installation:

Step-by-step installation instructions can be found here. Once properly installed, you should immediately notice an improvement.

CPU Utilization:

The sar utility from the sysstat collection gathers system activity numbers over time. The following sar snapshot taken from a dedicated Drupal server shows how the installation of MMCache can help reduce system load on even a heavily-optimized web

server (MMCache was installed around 12:00PM):

| 11:00:00 AM | CPU | %user | %nice | %system | %idle |
|-------------|-----|-------|-------|---------|-------|
| 11:10:00 AM | all | 3.71 | 0.00 | 1.85 | 94.44 |
| 11:20:00 AM | all | 3.86 | 0.00 | 0.50 | 95.64 |
| 11:30:00 AM | all | 4.49 | 0.00 | 0.33 | 95.18 |
| 11:40:00 AM | all | 4.05 | 0.00 | 0.36 | 95.58 |
| 11:50:00 AM | all | 3.76 | 0.00 | 0.36 | 95.88 |
| 12:00:00 PM | all | 3.38 | 0.00 | 0.52 | 96.11 |
| 12:10:00 PM | all | 0.52 | 0.00 | 0.10 | 99.38 |
| 12:20:00 PM | all | 0.79 | 0.00 | 0.20 | 99.01 |
| 12:30:00 PM | all | 0.57 | 0.00 | 0.12 | 99.31 |
| 12:40:00 PM | all | 0.59 | 0.00 | 0.12 | 99.29 |
| 12:50:00 PM | all | 0.44 | 0.00 | 0.11 | 99.45 |

Troubleshooting:

An easy way to tell if MMCache is working properly after following the installation instructions is to see if temporary files are being created in '/tmp/mmcache', or wherever you told them to be written with the 'mmcachecache.cache_dir' directive. If no files are appearing, something is wrong.

First, be sure that PHP has properly loaded mmcachecache. Create a short script on your web browser called 'phpinfo.php' as follows:

```
<?php
    phpinfo();
?>
```

Load that file in your browser to find a wealth of useful information. Search for any occurrences of the word 'MMCache'. If it's not there, then MMCache is not loaded. Double check your 'Configuration File (php.ini) Path' on that same page, and be sure that you modified the correct 'php.ini' file. Verify that you installed 'mmcachecache.so' into the directory specified by the 'extension_dir' directive. Also, try restarting your web browser to be sure the latest configuration changes have been made. Finally, be sure to look in your web server's error log to see if there are any hints there. (Note that the phpinfo() function call reveals a _lot_ of information about your system. For security reasons it is very unwise to make this information available to the general public. If you created 'phpinfo.php' in a public place, be sure to remove it when you're finished troubleshooting.)

Additional resources:

- Turck MMCache Home Page

Configuration

Drupal is an extremely flexible platform that provides you with many options for changing how your site looks, how users interact with it, and the kinds of information you wish to display. Although there are many configuration options, Drupal works well "out of the box" and there is very little initial configuration to perform. As your site evolves and your demands grow, Drupal makes it easy for you to dramatically alter its

look and add to its functionality. Best of all, because it is released under the GNU General Public License, Drupal is infinitely customizable and lets you tailor it to your specific needs.

Drupal's **basic settings** can be found on the settings page, which you can reach by clicking *administer Â» settings* in the navigation block. You must be logged in and have permission to access this page. After you change the settings, don't forget to click "Save configuration" at the bottom of the page.

The information you enter here is stored in Drupal's database to help Drupal decide how to prepare and serve your pages. Keep in mind that the settings on this page are only basic configuration settings. There are many other places to configure Drupal. These are discussed in the appropriate sections of this handbook.

A lot of the information you are asked to supply on the settings page is self-explanatory and there are brief help messages below each setting to guide you. Note that the "Slogan," "Mission," and "Footer message" may or may not be immediately visible, depending on the theme you are using. If you are new to Drupal, we recommend supplying all the information for these boxes. If you decide that you don't need or like how Drupal uses them, you can change it later (see the handbook section on configuring themes for more on this).

Settings

Below are links to provide you with explanations and tips for some of the more technical settings on the "settings" page. If you just want to get Drupal up and running, you can safely ignore these settings. You'll always be able to come back and experiment with them later. The one possible exception to this is the "file system settings" which you may wish to set now. For more information on the "File system settings," click its link below. This link can also help you resolve issues with errors that often appear at the top of the "settings" page after Drupal is freshly installed.

Anonymous user

Users who interact with your site without being logged in are labeled as "Anonymous" by default. Drupal gives you the option to change this to something different (e.g. "Anonymous coward"). The name you give anonymous users is used by Drupal when creating bylines for posts which typically reads something like, "Posted by Anonymous on January 1, 2006".

Default front page

This setting gives you control what Drupal-generated content a user sees when they visit your Drupal installation's root directory. For example, you might have created a node with a large collection of links to act as a table of contents to the different sections of your site and you want this directory to be what users see first.

This setting tells Drupal which URL users should be redirected to. It's important to note that the URL is relative to the directory your Drupal installation is in. So, instead of

"<http://www.example.com/node/83>"

or

"http://www.example.com/drupal_installation_directory/node/83,"

you need only type "node/83". For those not using clean URLs, note that there is no need to type in "?q=" before typing the URL.

By default, the "Default front page" is set to "node," which simply displays articles that have been "Promoted to front page." Note that when you change the "Default front page" to something other than "node", nodes that are "Promoted to front page" will no longer appear on the front page. They can however, still be viewed by visiting the relative URL path "node".

If you enter a value in here that is not a valid Drupal path, the user will be confronted with a "Page not found" error. You cannot redirect users to any web documents (e.g. a static HTML page) not created by your Drupal site.

Examples

Problem 1:

You want a particular node to be the first page user's see when the visit <http://www.example.com> (we are assuming Drupal is installed in the site's root directory).

Solution:

Determine the id number of the node you wish to have Drupal redirect users to. One way to determine the node's id number is to visit the node and look at the number after the last slash in your browser's address bar. This is your node's id number. Now set the "Default front page" to "node/id#". So, assuming your node's id is "83," you would type "node/83".

Problem 2:

You want user blogs to be the front page.

Solution:

Set the "Default front page" to "blog".

Problem 3:

You want content from a single category to appear on the front page

Solution:

Determine the category's id number. You can determine the id number by going to *administer > categories*. Roll the mouse on top of the "edit term" link next to the category to reveal the URL in your browser's status bar (usually located at the bottom of your browser's window). The number after the last slash in the URL is the category's id number. Now set the "Default front page" to "taxonomy/term/id#". If your category's id number is "5" for example, you'd write "taxonomy/term/5".

Clean URLs

By default, Drupal passes path arguments to itself via its internally generated URLs. This results in URLs that look like the following:

"<http://www.example.com/?q=node/83>." Notice the rather unsightly "?q=". Besides being unaesthetic and hard to read, it also stops many search engines from indexing your site.

You can tell Drupal to use "clean URLs", eliminating the "?q=" in internal URLs. Note that this works only for Apache servers which have mod_rewrite installed.

Warning. Enabling "Clean URLs" if your server is not properly configured can make it difficult to navigate back to administration pages to undo your mistake. If you find yourself in this situation, you can return to the administrative settings page by typing in the URL in the 'non-clean' form: <http://www.example.com/?q=admin/settings>.

Enabling clean URLs involves three steps:

1. Enable mod_rewrite for Apache. Please talk to your web host or consult the Apache documentation for mod_rewrite to get more information on how to do this. At a minimum, this will involve making sure that mod_rewrite is enabled for your installation of Apache. It will have to be either compiled-in or made available as a loadable module. In the latter case, make sure to tell Apache to load the module by including

```
LoadModule rewrite_module modules/mod_rewrite.so
```

in your Apache configuration file.

2. Edit your Apache configuration files for your site: the configuration information may be found in httpd.conf, a virtual-host-specific file, or in an .htaccess file in your Drupal installation directory. The main configuration option which may need to be changed for your site is the RewriteBase. For example, if your Apache DocumentRoot is /var/www/ (i.e., /var/www/index.html is what is displayed when you point your browser at <http://www.example.com/>) and your Drupal installation is installed in the subdirectory /var/www/mysite/, then the RewriteBase should be set to /mysite.
3. Finally, enable clean URLs on your administration/settings page. First, see if you can go to the settings page on your site using clean URLs: type in the URL <http://www.example.com/admin/settings> (where www.example.com is replaced by your hostname). If you get no errors and get to the same page you would have gotten to by clicking on "administer" then "settings", then you know that you have setup the ReWriteRule rules successfully and can then click the Yes checkbox for "Clean URLs".

Note: The standard Drupal installation contains a sample .htaccess file which supports clean URLs. It is easy to miss copying this file, because of the leading "dot".

Error pages

Drupal's page error messages are meant to be direct and to the point. If you want page error messages that are a little more user-friendly, Drupal allows you to customize them. Just follow these steps:

1. Create two nodes, one for each kind of page error (403 and 404).
2. Determine the id number of the node you wish to have Drupal redirect users to. One way to determine the node's id number is to visit the node and look at the number after the last slash in your browser's address bar. This is your node's id number.
3. Now enter the paths to your nodes into the appropriate boxes. For example, if the node id number for 403 error codes is "83," you would type "node/83" into the "Default 403 (access denied) page" setting.

Error reporting

If your site is actively serving pages on the Internet to users you may not trust, it's wise to set error reporting to "Write errors to the log". While this may make bugs and errors a little harder to catch, it makes your site more secure by hiding valuable information about your site from potential hackers.

You can tell Drupal to automatically discard error log entries older than the age you specify. This requires a crontab to be set up. See the discussion about crontab's elsewhere in this handbook.

Cache support

Busy Drupal sites may want to consider caching their pages to lighten the load on their server and speed up page generation times.

Normally, every time you visit a Drupal page, Drupal makes dozens of queries to the database to pull out the data needed to generate the HTML that your web-browser renders. On a large site with many modules installed or with lots of content on a page, the number of queries per page could rise into the hundreds. Usually, you don't notice all the work Drupal does because computers are very fast and Drupal is very efficient. However, on very busy sites with many hundreds or thousands of page views per minute, the amount of work required to serve each page may start to slowing the server to a crawl.

Busy sites can reduce the work required to generate pages by enabling Drupal's *page cache*.

With the cache turned on, Drupal stores all the HTML code for any page visited by an anonymous user directly into the database. When another request for the same page comes along, Drupal knows to fetch this page out of the database rather than re-generating it from scratch. The result is that hundreds of queries are replaced

with one single query, thereby significantly lightening the load on the web server.

File system settings

Drupal provides configuration settings to control whether, and how, users and administrators can upload files for use by Drupal.

Note: Unconfigured or improperly configured Drupal installations may display one or more error message at the top of the "settings" page, indicating that either the "Temporary directory" or "File system path" directories do not exist and/or their permissions are not set properly. Simply create these directories and set their permissions so that Drupal can write and read from the directory.

If you are unsure about where or how to create these directories or how to change their permissions, contact your web hosting service for further assistance.

Download method

There are two possible settings: "Public" and "Private".

Set to "Public" if:

You don't care if any user, even anonymous users, can download the files uploaded by other users.

Set to "Private" if:

You wish to restrict the ability of some users to download files uploaded by other users.

Notice the warning on the below this setting which reads "You can change this at any time, however all download URLs will change and there may be unexpected problems so it is not recommended." To avoid problems like this in the future, we recommend setting the "Download method" to "Private" since this still allows you to let all users download any file until you instruct Drupal otherwise.

Path settings

File system path

By default, this is set to "files". We recommend leaving this setting alone.

Temporary directory

By default this is set to "/tmp" which is the temporary directory common in GNU/Linux distributions. If you are using a Windows or other kind of server, we recommend setting it to "tmp" (no slash). Drupal will automatically create the temporary directory as a subdirectory of the "File system path."

Date and time settings

Drupal allows you to configure how dates and times are formatted and displayed. When making the format settings, you should probably consider the culture of your target audience. Below are suggested configurations for the "Default time zone" and "Configurable time zones" options.

For sites where most users live in a small geographic region:

Set the "Default time zone" to the time zone of the region and disable configurable time zones.

For sites where most users live in a region that spans a few time zones:

Set the "Default time zone" to the time zone usually considered to be the "standard" time zone and disable configurable time zones. For example, in the United States, you'd set your site's time to the timezone that corresponds to Eastern Standard Time.

For sites where users are likely to be scattered across the globe:

Set the "Default time zone" to the time zone to GMT (+0000) and enable configurable time zones.

Customizing the interface

When launching a new drupal site, here are some things you can do to personalize the design and architecture of your drupal site.

- **Choose a Theme.**

The look and feel of Drupal is primarily controlled by the theme you have applied to your site. A site can even have multiple themes. A good first step is to go to *administer > themes* and set a new theme as your default. You can find more themes on the download page after the list of modules. Once you download a new theme you will need to install it on your system.

- **Create your own Theme.**

Many Drupal sites will need a more unique look than these pre-built themes can offer. Therefore, many developers will want to write their own themes. Theme development requires a working knowledge of HTML/CSS and possibly some rudimentary PHP depending on the complexity of your theme.

- **Customize the Navigation.**

The menus that are displayed on the top and bottom of the page are configured in *administer > themes*. Select the *configure* tab and scroll down to *Menu Settings*. The primary and secondary links can be defined here, using straight HTML. If the primary links are left blank your navigation will be created based on your installed modules.

Each theme has an individual configuration page (listed at the top of the global settings page) as well. Unfortunately, if you are using a theme that uses the PHP Template theme engine, then your navigation must be defined in that theme's individual theme area (see : primary and secondary link functionality is retarded).

- **Customize Text Strings**

You can also change the text strings throughout drupal using the locale feature, which was designed for running drupal in different languages, you can personalize almost all of the text in drupal.

In fact, you can replace a string like "create blog entry" with html markup such as references to graphics.

Customizing user login

In the default setup the Drupal login block is always displayed unless a user is logged in. This may not always be desirable. For example, if you are using Drupal to create a site that has a very small number of people actually logging into the system to create or edit content, then you probably don't want a large portion of your screen real-estate taken up with a login block that doesn't relate to them. This also confuses the majority of your users that will not have the option to login.

To disable the login block:

1. Goto the block configuration (*administer » blocks*)
2. Deselect the check box for *User login* in the *Enabled* column

Your regular content editors and administrators can still login to the site by directly accessing the login page, <http://www.example.com/user>.

You can also create more customized login blocks.

Congestion control: tuning the auto-throttle

Overview:

This page will be of most benefit to Drupal users that have their websites hosted on a shared server. When you have little or no control over how your webserver is tuned, it can be extremely difficult to prepare for unexpected loads, such as a link from Slashdot or other high-traffic site. The steps below describe Drupal's built in congestion control mechanism, the auto-throttle, and walks you through its configuration.

Please note: if you have complete control over the server that's hosting your website, you should tune the operating system, database, webserver, and PHP for optimal stability and speed.

Tuning the auto-throttle:

1. Enable the Drupal cache:

One of the most dramatic performance improvements you can make with Drupal is by enabling cache support. This greatly reduces the overhead associated with displaying a page to an anonymous guest. To enable the cache, go to your site's main configuration page at *administer* → *settings*, scroll down to "Cache settings" and check "Enabled" under the words "Cache support:", then click "Save configuration".

2. Enable the statistics module:

To use the auto-throttle, you will first need to enable the statistics module. This can be found on your site at *administer* → *modules*. Put a check mark for statistics in the status column, and click 'Save configuration'.

3. Enable the access log:

Enable the statistics module under *administer* → *modules*. With the statistics.module enabled, you now must enable the access log, by clicking *administer* → *settings* → *statistics* and making sure you select "Enabled" under "Enable access log:". The access log writes an entry in a database table every time your site serves a page. The auto-throttle utilizes this information to monitor how much traffic is hitting your site. It is not important how long you retain access logs, so if you are only using this information for the auto-throttle and want to keep your database as small as possible, you can safely adjust this all the way down to 1 hour. When finished, click "Save configuration".

4. Enable the throttle module:

Now you need to enable the throttle module. Return to the module administration page on your site at *administer* → *modules* and put a check mark in the status column, then click 'Save configuration'.

5. Enable the "throttle status" block:

In order to properly tune the auto-throttle on your website, we need to have an understanding of how much traffic your site gets. The throttle module provides a block for this purpose. Go to the block administration page on your site at *administer* → *blocks*, and enable the "Throttle status" block. For this example, do not check "custom" or "throttle", but feel free to adjust the "weight" and "region" for whatever you prefer. Now click "Save blocks".

6. Configure "throttle status" block access permissions:

It is not desirable to allow your site's users to view the "throttle status" block that we have just enabled. This block is only intended as an administrative tool. If you are the only administrator of your site, and you administer your site as uid=1, then you don't need to set up any permissions -- by default, uid=1 has all permissions. However, if you have multiple administrators or administer your site with a different user account, you will need to click *administer* → *access control* and give your administrative group the "access throttle box" permission.

It is important to be aware that all users that are in this role will now see the "throttle status" block, as there is some overhead involved in displaying this block in the form of 1 database query per page displayed.

7. Enable the auto-throttle:

When you reload your website, you should now see the "Throttle status" block. All it says at this point is "Throttle: disabled", meaning that currently the auto-throttle is not turned on.

The auto throttle can be enabled on the throttle module administration page at *administer » settings » throttle*. You can quickly find this page by clicking the word "disabled" within your "Throttle status" block. On the resulting administration page, click "enabled" underneath the words "Enable auto-throttle", and click "Save configuration".

8. Monitoring the "Throttle status" block:

You should now notice that there is more information displayed in the "Throttle status" block. The only thing we're interested in at this time is the bottom section that should read something like, "This site has served 13 pages in the past minute." This means exactly what it says: during the past 60 seconds, your Drupal-powered website has served 13 pages to visitors of your site. This includes all pages that have been served, including pages viewed by yourself, registered users, anonymous guests, RSS clients, search engine spiders/bots, and cron.

To properly tune the auto-throttle, you will need to monitor this block carefully, getting a good feel for how busy your site is on average. In particular, you want to know how busy your site is on average at the busiest time of each day. In our example, we will assume that your site is serving between 10-12 pages a minute when it is busy. We will now use this information to tune our throttle.

9. Tuning the auto-throttle:

Once again return to the throttle module administration page at *administer » settings » throttle*. (A quick shortcut is to click on the word "enabled" within your "Throttle status" block.) We will now adjust the two options within the "Auto-throttle-tuning" section of this page.

The first option is the "Auto-throttle multiplier". It is generally a good idea to set your auto-throttle multiplier to a number nearest how many pages your site serves on average when it is busy. In our example, this was 12, so we will set the auto-throttle multiplier to "12 (0,12,24,36,48,60)". Each of the numbers in the parenthesis is a "throttle level", with 0 being level 0, 12 being level 1, 24 being level 2, and so on up to 60 being 5. Each of these numbers is a multiple of the number 12 that we have selected. When our site starts serving 12 pages a minute, the auto-throttle will adjust itself to be at level 1. Should the site become twice as busy as normal and start serving 24 pages in a minute, we move up to level 2. This continues until your site is serving more than 5 times a normal load (1 page per second), at which time the auto-throttle level will be set to 5.

The second option is the "Auto-throttle probability limiter". This fancily named configuration option is used to minimize the overhead of using the auto-throttle. You see, to calculate the current throttle level this module has to perform a database query. However, it turns out that one of the primary bottlenecks on a shared server is the database, so database queries are considered expensive. Thus, we adjust the "probability limiter" so that we perform our extra database query on only a certain percentage of page views. If you set this value to 10%, then we only perform the extra database query for approximately 1 out of every 10 pages displayed by your Drupal-powered site. It is unlikely you'd want to set this to anything higher

than 10%, however for busy sites you may wish to set it lower. Be aware that the lower you set this value, the longer your auto-throttle will take to detect a surge in load.

10. Auto-throttling blocks:

It is possible to configure blocks to be automatically disabled when the auto-throttle reaches a maximum level of 5, indicating that your site is currently experiencing a severe load. This could happen for a number of reasons, such as a link from Slashdot, or being indexed by sometimes over-aggressive googlebots, or even an intentional DoS (Denial of Service) attack. Under these heavy loads, you may find your site choking, usually reporting a MySQL error saying something like "Too many connections". By automatically disabling blocks, the cost of generating pages on your site will require less database queries and thus your site will be able to better withstand a greater number of hits.

To throttle blocks go to the block administration page on your site at *administer » blocks*. Now, for any blocks that should be disabled when your site is under a severe load click the "throttle" checkbox. It is recommended that you select nearly all boxes, except perhaps "Navigation" and "User login", as it will be rare that the auto-throttle actually causes them to be temporarily disabled. If you have an especially under-powered webserver, you may even wish to enable the throttle for the "User login" block so that users will be discouraged from logging in under heavy loads, as each page viewed by a user has to be dynamically built rather than displaying them from the cache.

Now, when your site comes under a heavy load, all blocks that have "throttle" enabled will be automatically disabled. As long as your site remains under a severe load, the blocks will remain disabled, optimizing your page and helping to prevent your database from choking. When the load starts to decline, the blocks will be automatically restored.

11. Auto-throttling modules:

It is also possible to configure entire modules to be automatically disabled when the auto-throttle reaches a maximum level of 5, once again indicating that your site is currently experiencing a severe load.

To throttle modules, go to the module administration page on your site at *administer » modules*. Now, for any module that should be disabled when your site is under a severe load click the "throttle" checkbox. Deciding which modules to throttle can be more difficult than deciding which blocks to throttle. It is recommended that you experiment in your development environment, disabling modules and seeing how this affects your site. The more underpowered your webserver, the more modules you will want to throttle. Generally speaking, throttling modules will usually have a larger affect than throttling blocks.

Now, when your site comes under a heavy load, all modules that have "throttle" enabled will be automatically disabled. All aspects of the module will be disabled, including any links, pages and/or blocks that the module may have generated. As long as your site remains under a severe load, the modules will remain disabled, optimizing your page and helping to prevent your database from choking. When the load starts to decline, the modules will be automatically restored.

12. Continuous auto-throttle tuning:

Your site is now potentially able to deal with heavier loads, however it is important to continue to monitor the "Throttle status" block to be sure you have properly configured your site. In particular, watch the "Current level" field, and be sure that under an average busy load you're not going above level 2. If you are, then you'll probably want to go back and adjust the "Auto-throttle multiplier" to a higher value as described earlier. On the other hand, if your site is always showing a "Current level" of 0, then you'll probably want to go back and adjust the "Auto-throttle multiplier" to a lower value.

Also note that over time your site's popularity may change. So, what was a perfect throttle setting a few months ago may be too high or too low this month.

13. Slashdotted!

Even with the auto-throttle enabled and configured, you may find that your server chokes when an extremely busy site such as Slashdot links to your site. It may be that your shared webserver simply can't handle the load, however don't give up too quickly. Here are a few more tips:

- Try adjusting the "Auto-throttle multiplier" to a lower value so your auto-throttle can detect a surge sooner.
- Try adjusting the "Auto-throttle probability limiter" to a higher percentage, again so that the auto-throttle is quicker to detect a surge. It is not advised to increase this value much beyond 10% however, as this may have a negative impact on your site's performance.
- Enable "throttle" for all but the absolutely essential blocks.
- Enable "throttle" for all but the absolutely essential modules. (this is perhaps the most significant suggestion)
- Hack your theme to be auto-throttle aware, automatically disabling large images when your site comes under a heavy load. Refer to the `throttle.module`'s "`throttle_status()`" function.
- Talk to your web host about how they can better tune your webserver...

Adding syndicated content (newsfeeds, RSS) to your site

Drupal has the ability to aggregate syndicated content (e.g. rss feeds) from multiple sources onto your websites. To learn more about this concept including such confusing terms as newsfeeds, rss, atom, and syndication, please read Drupal as a news aggregator for a more in-depth description of these services.

Add a new newsfeed

To add a newsfeed or syndicated content to your Drupal site:

1. Goto the modules configuration page (`administer > modules`), and enable the *aggregator* module
2. Once you have enabled the aggregator module you will be able to go to the aggregator configuration page (`administer > aggregator`).
3. Select the *add feed* tab.
4. Define the *Title*. This will be the heading of the newsfeed throughout the site.
5. Define the *URL* of the remote news feed, such as
`http://www.example.com/coolnewsfeed.rss`
6. Set the *Update interval*. Keep in mind that some news feeds are starting to install throttling software that may prevent you from accessing their feeds too often. Only pull content as much as you actually need to.
7. If you want the items for this news feed to have a block then you must change the *Latest items block* selection from it's default. The options control the number of items displayed in this block. Once you have selected this option you must

enable your newsfeed block in the blocks configuration page (*administer > blocks*)

If you leave the *Latest items block* setting at it's default your users will only be able to access the news feeds through the aggregator URL, <http://www.example.com/yoursite/aggregator> or <http://www.example.com/yoursite/aggregator/sources/1>, where 1 is the number of the newsfeed you have created.

Configuring newsfeed display

Newsfeeds can be displayed individually as blocks on your site, within categories which are also listed as blocks, or can be accessed through either method through the news aggregator link in the site navigation block.

- **Individual display as blocks**

This method is described in the setup above

- **Display of newsfeeds in category blocks**

Newsfeeds can be broken up into categories. To create a category:

1. Goto the newsfeed config page (*administer > aggregator*) and select the *add category* tab.
2. Provide a *Title* and *Description*.
3. To make news feeds defined to this category appear as a block select a number of items from the *Latest items block* pull-down.
4. Enable this new block on the block configuration page (*administer > blocks*), then return to the newsfeed config page (*administer > aggregator*).
5. We will now need to add one or more news feeds to this category. Select the *List* tab and select the *edit* link in the row of the newsfeed item you would like to add to this category.
6. Under *Automatically file items* select the checkbox next to the category you just created. Now when new news items come in they will be displayed in the category block. The power of this feature is being able to bring multiple news feeds together under one category.
7. If you already have news items updated on your site the category will not index them. To get the old news items into the new category, click the *list* tab, click *remove items* from the newsfeed you want, and then click *update items*. This will update the items and index them into your new category.

- **Display under news aggregator link**

When you enable the aggregator module, drupal creates a link in your site navigation. Under this section your users can view all individual news feeds and news feed categories you have created.

Relevant URLs in this section are:

- <http://www.example.com/yoursite/aggregator> - news aggregator home
- <http://www.example.com/yoursite/aggregator/sources/#> - index of individual newsfeeds where the number indicates the particular feed
- <http://www.example.com/yoursite/aggregator/categories/#> - index of newsfeed categories where the number indicates the category

If you are looking for content, there are several sites that provide an index of news feed services on the internet including:

- Syndic8
- Yahoo RSS Feeds

Database table prefix (and sharing tables across instances)

Some web hosts limit their customers to one database. Thus, no duplicate table names are possible. In order to assure that these admins can still use Drupal, and even use multiple installations of Drupal, Drupal offers *table prefixing*.

In order to use this feature, you must currently edit the script `database/database.x` in order to create tables prefixed by the string of your choice. For example, change all statements from the format of

```
CREATE TABLE access
to
CREATE TABLE dr1_access.
```

Then use `dr1_` (for example) as value of `$db_prefix` in your `sites/example.com/settings.php` file.

Note: The following tables contain data that is highly site specific and should therefore **not** be shared:

- cache
- variable

Advanced usage

Table prefixing may be optionally applied to some tables and not others. This has the effect that multiple Drupal installations can share common tables. One interesting application for this is to share the taxonomy tables (vocabularies, term_data). Another interesting use is to share *users* across Drupal installations.

In order to use this capability, create two drupal installs in same DB using different database prefixes. In this example, one is prefixed 'master_' and the other 'content_only_'. Then edit the conf.php file of 'content_only_' so that it points some tables to the 'master_'. For sharing users, add the following:

```
$db_prefix = array(
    "default" => "content_only_",
    "users" => "master_",
    "sessions" => "master_",
    "role" => "master_",
    "authmap" => "master_",
    "sequences" => "master_"
);
```

Super advanced usage

[tested only on mysql so far] It is possible to keep the multiple Drupal installations in different databases but still share common tables. To do this, specify the database name as part of the prefix. For example,

```
$db_prefix = array(
    "default" => "content_only.",
    "users" => "master.",
    "sessions" => "master.",
    "role" => "master.",
    "authmap" => "master.",
    "sequences" => "master."
);
```

In the example above, *content_only* and *master* are **databases**.

Helping search engines index your site

Drupal by itself is very search engine friendly. For example it is not uncommon for Drupal based sites to have a Google ranking of 5 or higher (out of 10) where using the same content on another CMS would score much lower.

Still, you can make Drupal even more search engine friendly by changing some default parameters. There are several Drupal settings you can tweak to make Drupal even more search engine friendly.

1. First of all you might want to enable friendly URL's
2. Then, make sure that you get rid of the session ID in the URL by changing the .htaccess if you are using version 4.5.x. On 4.6, session IDs in URLs are disabled by default.
3. Optionally, use URL aliasing for some or all nodes. You can use the pathauto module to automatically create aliases for new nodes.

Controlling what gets indexed -- the robots.txt file

By default Drupal does not ship with a `robots.txt` file. This file is the mechanism almost all search engines use to allow website administrators to control what will be indexed.

By adding this file to the root of your (virtual) webserver, you can quide the search engines' "bots" through your site or forbid indexing parts of your site. See for an example the file for drupal.org itself at <http://drupal.org/robots.txt>.

If you want to have `robots.txt` file, please follow the instructions below. For more details check <http://www.robotstxt.org>

Create a file with the content as shown below and call it "robots.txt"

```
# small robots.txt
# more information about this file can be found at
# http://www.robotstxt.org/wc/robots.html
# lines beginning with the pund ("#") sign are comments and
# can be deleted.
# if case your drupal site is in a directory
# lower than your docroot (e.g. /drupal)
# please add this before the /-es below
# to stop a polite robot indexing an exampledir
# add a line like (delte the #'s)
# user-agent: polite-bot
# Disallow: /exampledir/
# a list of know bots can be found at
# http://www.robotstxt.org/wc/active/html/index.html
# see http://www.sxw.org.uk/computing/robots/check.html
# for syntax checking
User-agent: *
Crawl-Delay: 10
Disallow: /aggregator
Disallow: /tracker
Disallow: /comment/reply
Disallow: /node/add
Disallow: /user
Disallow: /files
Disallow: /search
Disallow: /book/print
```

This file tells indexing robots that they should avoid pages that contain content for users only, for example the search page, or the 'Add a comment' forms for nodes.

Many robots obey the "Crawl-delay:" parameter. Since Drupal sites seem to be popular with search engines and lots of people have more aggressive bots than visitors at their site, it might be wise to slow down the robots by adding a robots.txt line like this:

```
User-Agent: *
Crawl-Delay: 10
```

Here 10 is the delay in seconds between page requests.

Both "Slurp" (The robot that is indexing for yahoo and altaVista) and the Microsoft robots for the MSN sites obey this parameter. Googlebot does not use the "crawl-delay" parameter yet but will likely do so in an upcoming version.

Change the file as you wish and save it. Now upload it to your webserver and make sure it is in the root of the (virtual) webserver. If you have installed Drupal in a subdirectory (for example /drupal), then change the URL's in the robots.txt file but place the file in the root of the webserver, not in the root of your drupal installation.

Now watch the robots visit your site and after some time, monitor your log files ("referrer log") to see how many visitors came from a search engine.

PHP page snippets: putting dynamic content into your main page

Once Drupal is installed you will notice an option under **CREATE CONTENT** to create a **PAGE**. It is a standard out-of-the-box drupal feature and when creating a **PAGE** you have 3 filter options when submitting i.e. Filtered HTML, Full HTML or PHP.

The PHP Snippets below are intended for use within a drupal page filtered as PHP that simply enables you to "pull" specific content from your drupal database. Allowing you to create dynamic and sophisticated page layouts such those found at busy portals, such as www.mtv.com or www.yahoo.com.

Make sure you check the custom block examples as well.

- Very simple to use and implement. Copy n paste snippets into your page
- Allows users to insert "block style" content in the main page
- build simple or sophisticated main page layouts with dynamic content
- pull specific content from your drupal database to insert into a page

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

How to insert and use the PHP Snippets in your pages

The PHP SNIPPETS come with a brief introduction on what it does and if it only works with specific versions of Drupal.

Some are very similar to how you **SETUP CUSTOM BLOCKS WITH CONTENT THAT APPEAR IN THE SIDEbars** but the snippets below are intended for use within your main NODE page as opposed to your sidebars.

Once you get used to how the PHP Snippets work and if you are familiar with how to setup up a HTML table or using DIVs, you can mix snippets together in the same page creating sophisticated layouts like those found at <http://www.mtv.com> or <http://www.yahoo.com>.

1. Go to CREATE CONTENT --> PAGE
2. Give your page a TITLE
3. In the BODY text area, paste in your PHP Snippet. Make sure you include the opening and closing PHP statements and remove any extra spaces, line breaks after the final ?> which tells drupal that this is the end of the php page.
4. Select the PHP CODE filter.
5. Click SUBMIT or PREVIEW to view your page.

More sophisticated layout/styling of node pages

The following is intended for people not familiar with PHP coding and illustrates how to apply styling to content inserted into a page using PHP.

Here is an example of a very simple PHP snippet that displays the Site slogan.

```
<?php
/**
 * the following displays the site slogan
 */
print variable_get("site_slogan", "");
?>
```

Click for a simple example of how to add styling to the content displayed by the php snippet.

Using some simple HTML functions, such as creating a table or using DIVS, you can start creating much more sophisticated layouts using multiple snippets in the same page. For some guidance and tips it is worth looking at a more sophisticated php Snippet that inserts the site mission and a list of upcoming events in a 2 column table side by side, followed by a list of the recent weblog entries.

```
!-----!
!-----!
!----site---!--upcoming----!
!---mission---!--events----!
!-----!
!-----!
!-----!
!-----!
!-----recent weblog entries----!
!-----!
!-----!
```

How to use the PHP Snippets with the front_page.module

USING PHP SNIPPETS WITH THE FRONT_PAGE MODULE

Probably the most common use of php snippets will be the front page of your site.

To use any of these snippets using the **front_page.module** (allows you to specify a "splash" page to your site and different front pages for anonymous/authenticated users) Simply follow these steps once you have the front_page.module installed properly:

Step 1: Go to ADMINISTER --> SETTINGS --> FRONT_PAGE

Step 2: In the teaxt areas available, paste in your PHP Snippet(s).

Step 3: Select the ALLOW EMBEDDED PHP option.

Step 4: Select if you want a FULL/THEMED Front Page.

Step 5: Click on SAVE CONFIGURATION once you are happy with your front_page settings.

A guide to submitting your own PHP snippets

Please post your own snippets up here for others and follow these simple guidelines.

1. Click on ADD CHILD PAGE from the main page in this section (PHP PAGE SNIPPETS: PUTTING DYNAMIC CONTENT INTO YOUR MAIN PAGE).
2. Give your snippet a short and obvious title so it is easy for others to see what your snippet does.
3. Please include a simple introduction at the top of your PHP Snippet which explains:
 - What the snippet does
 - Which versions of Drupal has it been tested with
 - Any extra information for newbies that you think is relevant

Countdown (x) days to a specific date and display a dynamic message

PLEASE NOTE! The following snippet is user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * This php snippet displays (x) days left to a specific event
 *
 *
 * Change the values for keyMonth, keyDay and keyYear to suit
 *
 *
 * Tested and works with drupal 4.6 and 4.5
 */
$keyMonth = 7;
$keyDay = 14;
$keyYear = 2005;
$month = date(F);
$mon = date(n);
$day = date(j);
$year = date(Y);
$hours_left = (mktime(0,0,0,$keyMonth,$keyDay,$keyYear)) -
time())/3600;
$daysLeft = ceil($hours_left/24);
$z = (string)$daysLeft;
if ($z > 1) {
print "There are <font size=\"4\" color=\"red\>" ;
print $z;
print "</font> days left until whatever happens</p>" ;
}
?>
```

Create a list of node titles of a specific type, within certain dates/times

```
<?php
/**
 * Creates a list of node titles of a specific type, within
certain dates
 * with a link to each node.
 *
 * To change which type is listed, simply edit the $node_type
string.
```

```

* To change the starting date, simply change the $start_stamp.
* To change the ending date, simply change the $end_stamp.
*
* This works with drupal 4.6
*/
$node_type = "image";
$start_stamp = 'June 9 2005';
$end_stamp = 'June 10 2005';
$start_stamp = strtotime($start_stamp);
$end_stamp = strtotime($end_stamp);
$sql = "SELECT node.title, node.nid FROM node WHERE
node.created < $end_stamp AND node.created > $start_stamp AND node.type
= '$node_type' ";
$output .= "<ul>";
$result = db_query($sql);
while ($anode = db_fetch_object($result)) {
$output .= "<li>".l($anode->title,
"node/$anode->nid")."</li>";
}
$output .= "</ul>";
return $output;
?>
```

display (x) random thumbnails in a page

PLEASE NOTE! The following snippet is user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```

<?php
/**
* This php snippet displays (x) random thumbnails with a link
* to
* the full images in a page.
*
* (assumes you already have the IMAGE MODULE installed)
*
* To increase/decrease the number of thumbnails listed
* change the $thumbs field to suit.
*
* Tested and works with drupal 4.6
*/
$thumbs = 0;
while ($thumbs<10) {
$images = (image_get_random($count = 1, $tid = 0));
print l(image_display($images[0]),
```

```

'thumbnail'), 'node/'.$images[0]->nid, array(), null, null,
FALSE, TRUE);
$thumbs++;
}
?>

```

Display a list of (x) most recent weblog entries

PLEASE NOTE: The php snippets are user submitted and it is impossible to check every one, so use at your own risk. For users who have setup drupal using an alternate database to the default, please note that the snippets may contain some database queries that may or may not work.

```

<?php
/**
 * the following displays a list of the 10 most recent weblog
titles
* and links to the full weblogs. If you want to
increase/reduce
* the number of titles displayed..simply change $listlength
value
*
* This php snippet works with drupal 4.6.
*
*/
$listlength="10";
$output =
node_title_list(db_query_range(db_rewrite_sql("SELECT n.nid,
n.title, n.created FROM {node} n WHERE n.type = 'blog' AND n.status = 1
ORDER BY n.created DESC"), 0, $listlength));
print $output;
?>

```

Display a list of (x) node titles of a specific type

```

<?php
/**
* Creates a list of node titles of a specific type
* with a link to each node.
*
* To change which type is listed, simply edit the $node_type
string.
* To change the number of node titles listed, simply edit the
$list_no number.
*
* This works with drupal 4.5 and drupal 4.6
*/

```

```

$node_type = "flexinode-1";
$list_no =5;
$sql = "SELECT node.title, node.type, node.nid FROM node
WHERE node.type = '$node_type' LIMIT $list_no";
$output .= "<ul>";
$result = db_query($sql);
while ($anode = db_fetch_object($result)) {
$output .= "<li>.".$anode->title,
"node/$anode->nid")."</li>";
}
$output .= "</ul>";
return $output;
?>

```

Display a list of (x) upcoming events in a scrolling box without javascript

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```

<?php
/**
 * The following snippet displays the next 10 upcoming events
 * in a scrolling box that is set in a simple DIV styled
 * inline.
 * To increase or decrease the number of events listed just
 * change
 * the $listlength value
 *
 * no javascript is required and the current
 * height, width is 150 X 320 pixels.
 * to increase decrease that simply change the values for
 * the $boxheight and $boxwidth
 *
 * works with most popular browsers
 * Tested in IE6, Mozilla Firefox 1.0 & 1.5, and NN7
 *
 * Tested and works with with drupal 4.5.x and Drupal 4.6.x
 *
 */
$listlength="15";
$boxheight="150px";
$boxwidth="320px";
print "<h1>Upcoming Events</h1><div
style=\"unicode-bidi:bidi-override; direction:rtl; display:block;

```

```

width:$boxwidth; height:$boxheight; overflow:auto; padding-left:10px;
border:1px solid #ba8; \">><div dir=\"ltr\">><p>";
print event_block_upcoming($limit=$listlength);
print "</p></div></div>";
?>
```

Display a list of a certain content type, with teasers

```

<?php
/**
 * This php snippet displays content of a specified type, with
teasers
*
* To change the type of content listed, change the
$content_type.
*
* Works with drupal 4.6
*/
$content_type = 'story';
$result1 = pager_query(db_rewrite_sql("SELECT n.nid,
n.created FROM {node} n WHERE n.type = '$content_type' AND n.status = 1
ORDER BY n.created ASC"));
while ($node = db_fetch_object($result1)) {
  $output .= node_view(node_load(array('nid' =>
$node->nid)), 1);
}
print $output;
?>
```

Display a list of category titles with links to the full term

Please note: These snippets are user submitted. Use at your own risk. For users who have setup Drupal using a database other than the default (MySQL), please note that the snippets may contain some database queries specific to MySQL.

```

<?php
/**
 * Creates a list of category titles with a link to each
category term.
* And the number of items in each term in brackets. (x)
*
* Category term title (x) and link to full term
* date of last update
*
*
* This works with drupal 4.6 and 4.5.
*
```

```

* If you improve this snippet please post a new comment
below.
*
*/
$result = db_query("SELECT d.tid, d.name, MAX(n.created) AS
updated, COUNT(*) AS count FROM {term_data} d INNER JOIN {term_node}
USING (tid) INNER JOIN {node} n USING (nid) WHERE n.status = 1 GROUP BY
d.tid, d.name ORDER BY updated DESC, d.name");
$items = array();
while ($category = db_fetch_object($result)) {
    $items[] = l($category->name . ' (' . $category->count
.' )', 'taxonomy/term/'. $category->tid) . '<br />' . t('
%time ago', array('%time' => format_interval(time() -
$category->updated)));
}
print theme('item_list', $items);
?>

```

Display a list of node titles from a specific category

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```

<?php
/**
 * Creates a list of node titles from a specific category
 * with a link to each node.
 *
 * To change which taxonomy is listed, simply edit the
 $taxo_id number.
 * To change the number of node titles listed, simply edit the
 $list_no number.
 *
 * This works with drupal 4.5 and drupal 4.6
 */
$taxo_id = 1;
$list_no = 5;
$sql = "SELECT node.title, node.nid FROM node INNER JOIN
term_node ON node.nid = term_node.nid WHERE term_node.tid = $taxo_id
LIMIT $list_no";
$output .= "<ul>";
$result = db_query($sql);
while ($anode = db_fetch_object($result)) {
$output .= "<li>" . l($anode->title,
"node/$anode->nid") . "</li>";
}

```

```
$output .= "</ul>";
return $output;
?>
```

Display a list of the next (x) upcoming events

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * the following displays a list of the 10 upcoming event
 * titles and
 * links to their full event nodes. To increase/decrease
 * the number of titles displayed..simply change the
 * $listlength value
 *
 * Works with drupal 4.5.x and drupal 4.6
 */
$listlength="10";
print event_block_upcoming($limit = $listlength);
?>
```

Display different page content to anonymous and authenticated users

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * The following simple snippet
 * displays different information to anonymous/logged in users
 * within a page.
 *
 * This works with drupal 4.5 and drupal 4.6
 */
global $user;
if ($user->uid) {
    return "This message is only visible for logged-in
users.";
}
if (!$user->uid) {
    return "This message is only visible for not-logged-in
users.;"
```

```
}
```

Display the (x) most recent nodes in full from a specific category

PLEASE NOTE! The following snippet is user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * This php snippet displays the most recent node in full
 * from a specific category
 *
 * To increase/decrease the number of nodes listed
 * change the $list_length value to suit.
 *
 * Works with drupal 4.6.x & 4.5.x
 *
 * Snippet submitted by Robert Garrigos (robertgarrigos)
 */
$taxo_id = 10;
$list_length = 1;
$sql = "SELECT * FROM node INNER JOIN term_node ON node.nid =
term_node.nid WHERE term_node.tid = $taxo_id ORDER BY node.created DESC
LIMIT $list_length";
$result = db_query($sql);
while ($anode = db_fetch_object($result)) {
$output .= theme('node', $anode, $teaser = TRUE, $page =
FALSE);
}
print $output;
?>
```

Display the (x) most recent weblog entries from a specific user

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * the following displays a list of the 10 most recent weblog
 * titles
 * and links to the full weblogs of a certain user.

```

```

* If you want to increase/reduce
* the number of titles displayed..simply change the
$listlength value
*
* for a different user change the $userid value
*
* works with drupal 4.6.
*
*/
$listlength="10";
$userid="8";
$output =
node_title_list(db_query_range(db_rewrite_sql("SELECT n.nid,
n.title, n.created FROM {node} n WHERE n.type = 'blog' AND n.uid =
$userid AND n.status = 1 ORDER BY n.created DESC"), 0,
$listlength));
print $output;
?>
```

Display the (x) most recent weblog entries with teasers & info.

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```

<?php
/**
* This php snippet displays the 10 most recent weblog entries
* with
* teaser & info.
*
* To increase/decrease the number of weblogs listed
* change the $listlength field to suit.
*
* Works with drupal 4.6
*/
$listlength=10;
$result1 = pager_query(db_rewrite_sql("SELECT n.nid,
n.created FROM {node} n WHERE n.type = 'blog' AND n.status = 1 ORDER BY
n.created ASC"), variable_get('default_nodes_main',
$listlength));
while ($node = db_fetch_object($result1)) {
  $output .= node_view(node_load(array('nid' =>
$node->nid)), 1);
}
print $output;
```

```
?>
```

Display/hide content from a specific IP address within a page

PLEASE NOTE These snippets are user submitted. Use at your own risk. For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * this Snippet allows you to display content ONLY to a
 * visitor from
 * a specific IP addresses in a page.
 *
 * Change the $allowed value to the IP address you want.
 *
 * This works with drupal 4.5 and drupal 4.6
 */
$allowed = '100.100.100.100';
$userip = $_SERVER['REMOTE_ADDR'];
if($userip == $allowed){
    print "This content can only be viewed by the IP address
you specify.";
}
?>
```

To reverse the situation and HIDE content to a user from a specific IP address

```
<?php
/**
 * this Snippet allows you to hide content from a visitor at
 * a specific IP addresses.
 *
 * Change the $hidden value to the IP address you want.
 *
 * This works with drupal 4.5 and drupal 4.6
 */
$hidden = '100.100.100.100';
$userip = $_SERVER['REMOTE_ADDR'];
if($userip != $hidden){
    print "This content is displayed to everyone except the
person from the IP address you specify.";
}
?>
```

Insert a quicklist of recent forum topic titles and links

PLEASE NOTE! The following snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * Creates a quicklist of recent forum topic titles and a link
 * to their
 * node.
 *
 * works and tested with 4.6
 * does not work with 4.5
 *
 * To increase the length of the list, change $listlength
 *
 */
$listlength="10";
$sql = "SELECT n.nid, n.title, l.last_comment_timestamp,
l.comment_count FROM {node} n INNER JOIN {node_comment_statistics} l ON
n.nid = l.nid WHERE n.status = 1 AND n.type='forum' ORDER BY
l.last_comment_timestamp DESC";
$sql = db_rewrite_sql($sql);
$content = node_title_list(db_query_range($sql, 0,
$listlength), t('Active forum topics:'));
if ($content) {
    $content .= '<div class="more-link">' . l(t('more'),
'forum', array('title' => t('Read the latest forum
topics.'))) . '</div>';
}
print $content;
?>
```

Insert an image before the promoted nodes on the frontpage

The front_page module allows for the display of a customized front page. This PHP snippet keeps the original front page (containing nodes promoted to the front page) and inserts an image (or other static content) in front of it.

Prerequisite: front_page module must be installed and configured properly. Go to ADMINISTER - SETTINGS - FRONT_PAGE, check "Allow embedded PHP code" and enter the PHP snippet into the "Front page HTML" box.

```
<center></center>
<?php
    print node_page_default();
?>
```

Tested with drupal version 4.6.0.

Insert the most recent poll

PLEASE NOTE! The PHP snippets are user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * the following displays the most recent poll
 * and assumes you have the poll.module enabled
 *
 * Works with drupal 4.6
 * Does not work with 4.5.x
 */
$sql = db_rewrite_sql("SELECT MAX(n.created) FROM {node} n
INNER JOIN {poll} p ON p.nid = n.nid WHERE n.status = 1 AND p.active =
1 AND n.moderate = 0");
$timestramp = db_result(db_query($sql));
if ($timestramp) {
    $poll = node_load(array('type' => 'poll', 'created' => $timestramp, 'moderate' => 0, 'status' => 1));
    if ($poll->nid) {
        // poll_view() dumps the output into $poll->body.
        poll_view($poll, 1, 0, 1);
    }
}
print $poll->body;
?>
```

maintenance redirect: automatically redirecting everyone but you away from the drupal site

PLEASE NOTE! The following snippet is user submitted. Use at your own risk! For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```
<?php
/**
 * This php snippet redirects everyone but you away from the
```

```

drupal site
* to a htm page. Useful for "undergoing maintenance" type
occasions
*
* put this snippet at the very top of your INDEX.PHP file in
your root drupal folder
* or in your front_page settings page and place your
closed.html
* file in the same folder.
*
* change the your_ip address to be your IP address so you can
still access the site
*
*/
$your_ip = '100.100.0.1';
if ($_SERVER['REMOTE_ADDR'] != $your_ip) {
  header('location:closed.html');
}
?>

```

Printing PHP variables from GET or POST forms

PLEASE NOTE These snippets are user submitted. Use at your own risk. For users who have setup drupal using an alternate database to the default (MYSQL), please note that the snippets may contain some database queries specific to MYSQL.

```

<?php
/**
* this Snippet illustrates how you display the values of a
submitted form
* in a Drupal page.
*
* Submitted by puregin
*
* This works with drupal 4.5 and drupal 4.6
*/
print "variable1: " . $_POST["variable1"] . "<br/>";
print "product: " . $_POST["product"] . "<br/>";
print "size: " . $_POST["size"] . "<br/>";
?>

```

NOTES:

Change `$_POST` to `$_GET` if you used GET as your form method or if you are using a URL to set the variables. (e.g., `example.com/page.php?variable1=hello`)

Using more than one php snippet in the same node (or front_page)

In some circumstances, if you are using two or more php snippets in the same node - or on your front_page.module page - you need add the following line at the **beginning of each php snippet you intend to use**

```
<?php  
unset ($output);  
?>
```

why?

The reason is to avoid the problem of using the `return $output;` more than once in the same node. The `return $output` command will *end* the reading of code and all subsequent code or text in that node (or front_page) will be ignored.

To avoid potential problems, rather than `return $output;`, you can use...

```
<?php  
print $output;  
?>
```

...for the last line of code.

But...

With each subsequent php-generated list you create in that same node (or front_page), you will get a *cumulative* list -- the content generated by each additional php snippet will be added to the `$output` list. (For example, the second php snippet's list of posts on "apples" would be *added* to the first php snippet's list of posts on "oranges," when what you really want is just a list of posts on "apples.")

So how would one "clear" the output results so one can have another php snippet creating output? **Start each of your subsequent snippets of php code with this:**

```
<?php  
unset ($output);  
?>
```

This will clear the previous results so that your 2nd, 3rd ... *nth* list will have unique results relevant to its own query.

Thus you will be able to have several php-generated outputs on the same page (such as lists of recent posts in several different taxonomy terms) without conflict or intermingling of results.

Redirecting your host.example.com to a specific page

It can be handy to have some subdomains for different purposes leading towards differnt pages on a website. A usefull subdomain might for example be downloads.example.com. Any user will understand what one can expect at such a domain.

Things like this can be done via a name behind the domainname with drupal 4.3.x in the core or in 4.2 with the path module. In that case you can point your users to example.com/downloads and this page drupal will rewrite towards the download page. The way of rewriting these domains is rather neat, but having subdomains in stead of filenames might in certain case have a better image or be more usefull.

The site drupal.org uses the shortcut hostname.example.com as well. For example to the documentation place, <http://documentation.drupal.org> leads to <http://www.drupal.org/node.php?id=253>.

So how can you use this service? What follows is a short howto.

Say you own the domain example.com (rfc2606);

Say you have a drupal page example.com/node/view/nid

You want to have a shortcut to this page labelled nid.example.com

Now you have to do the following things:

1. Edit your DNS
2. Edit your .htaccess

Regarding 1), editing your DNS

How DNS works wont be described here , but if you have basic knowledge of DNS and administer one or more domains, you should be able to add a hostname in your zone called nid(.example.com.). Please make sure you update the TTL.

Another way of doing this is by adding a wildcard in your domain. This wildcard (*) makes that any hostname in your domain will automatically be resolvable. Now wildcards for toplevel domains might be considered very bad (see the rants against verisigns sitefinder service), having one in a domain might be considered bad as well.

Note that there is one very big disadvantage of having all hostnames resolved. Any word (*and we do mean any word*) will point to your site, wether you like the word or not. Together with the standard drupal behavior of not generating an error pages (e.g. a 404 page), this can lead towards very undesired results. For example <http://i-think.drupal.org/rocks> will give a valid page, as well as less flatering ones...

Regarding 2) edit hour .htaccess

Once you edited your zone with the new hostname and you tested from various places that this hostname resolves towards the right IP adress, you can edit your

.htaccess file in the root of your webserver. This assumes you are using apache.

First make sure that you have mod_rewrite enabled in your apache webserver. There are more ways to make sure you have mod_rewrite as a module enabled in your webserver, an easy way is to make a small php file somewhere in your document root with phpinfo in it

```
<?php
phpinfo();
?>
```

Now surf towards this php file and search for *mod_rewrite* in the page. If this module is not enabled, ask your hosting party or download the module yourself and enable it.

Once you have mod_rewrite installed we can edit the .htaccess file in the root of your webdirectory. Start your favourite editor and check if the line

```
RewriteEngine On
```

is there. If not, add the line.

Now add the following entries:

```
RewriteCond %{HTTP_HOST} ^nid\.example\.com$ [NC]
RewriteRule ^(.*)$ http://example.com/node/view/nid [R=301,L]
```

Surf towards your new subdomain nid.example.com and see if you are going to example.com/node/view/nid. Check the error log in case things fail, you don't even have to restart the webserver.

Having shortcuts this way can be very handy for taxonomy

.

Hope you enjoy your new shortcut! Have fun.

The tolerant base URL

Instead of using a hard coded domain as your \$base_url in the includes/conf.php file, you might want to use

```
<?php
$base_url = 'http' . (isset($_SERVER['HTTPS']) ? : '');
$_SERVER['HTTPS'] == 'on' ? 's' : '' : '');
$base_url .= '://' . $_SERVER['HTTP_HOST'];
if ($dir = trim(dirname($_SERVER['SCRIPT_NAME']), '\,/')) {
    $base_url .= "/$dir";
}
?>
```

This has the advantage that whatever domain the user used to get to the site, he will maintain throughout his session.

Warning

- Email notifications may be issued under the domain which is used by the poster. If you access your site using `http://localhost`, you could send emails with that invalid URL. The only module which behaves this way today that I know of is `subscription.module` from Contrib.

Blocks

Blocks are the boxes visible in the sidebar(s) of your Drupal website. Most of the blocks that you will see (e.g., recent forum topics) are generated on-the-fly by various Drupal modules, but you can also create your own blocks.

Whether, and where, a given block will appear on a page depends on both the theme enabled and on administrative block settings. Block settings are controlled from the block administration screen, which you can reach by clicking `administer > blocks`. From this screen, you will be able to control whether each block is enabled where it will be placed on the page, and which pages it should appear on.

Block configuration

Whether a block appears on a page depends on a number of factors.

- First, the block must be *enabled* by checking its "enabled" box on the block management page;
- Second, the block's *custom visibility* settings must permit the block to be displayed for a given user. You can configure a block to appear always; to be displayed by default unless disabled by individual user preferences, or to be disabled by default but visible if enabled by individual users. Individual users can select or disable optional blocks from their "my account" page;
- Next, whether the block appears on pages for a given section of the site can be configured by the administrator by setting the block's *path*;
- The block can be configured to appear only for certain *content types*;
- Finally, if the *throttle module* has been enabled, and the block has its throttle box checked on the block management page, then it will only appear if the site's traffic is below a given threshold.

Once you have configured a block to appear as required on specific pages and in the correct column, you may wish to adjust the block's **weight** to position the block vertically within its column. A block's weight is set via a pull-down selector on the block management page. Heavier blocks (those with a more positive weight) "sink" to the bottom of a column, while lighter blocks "float" towards the top.

Restricting blocks to certain pages

Below those options, you will see a large text box. On each line, you can specify which pages that block will appear on -- or not appear on, depending one what you selected above.

The following are some annotated examples. They all assume that you choose the second option above; if you chose the first option, blocks will appear on all pages **other** than what you specify below. You're welcome to add more than one listing, each separated by lines. (Note that you cannot specify pages on which blocks appear as well as pages on which blocks do not appear. It is either one or the other.)

| | |
|------------|---|
| <front> | Specifies that the block appears on the front page. |
| node/42 | Specifies that the block will appear on the item with the node ID of 42. |
| aggregator | Specifies that the block appears on just the aggregator. |
| blog/* | Specifies that all URLs that start with blog/ will show the block. Note: if you want the block(s) to also appear on the main blog page, you need to add blog (that is, without the trailing slash) as well. |

Another example, if you have HTML, Javascript or PHP blocks you don't think are necessary for administrators to see, you can select "Show on every page except the listed pages." and use the following wildcard for the block:

- admin/*

Administrators can select which content type blocks appear on as well. This can be found under "Content specific visibility settings" after clicking the "configure" link for an individual block. Choose which content type blocks appear on by checking the box next to that content type. You can use content type-specific blocks in conjunction with the above restrictions based on URLs.

Example: preventing a block from appearing

I'm baffled....I have reviewed the Drupal guide on the site and read through numerous postings from the archive of this list, and have tediously tested many alternate paths.

I currently have this path (which works)

```
<^(?!\event)>
```

it used to to keep certain blocks off the month view of the calendar.

I want to add to that string something that does not display a block on a specific page (e.g. node/635)

I tried

```
<^(node/635) |^(?!event)>  
(did not work)
```

```
<( ^node/635) |^(?!event)>  
(did not work)
```

I'm not sure I am using the vertical bar | correctly. I had trouble finding examples of multiple paths.

I am also not specifying the node/635 correctly, as I could not get to work even when I removed the second part

```
^(?!event)
```

Thanks,
Andrew

Hi Andrew,

I can empathize with your situation! This is one of the most powerful, but also one of the most difficult, aspects of the block mechanism in Drupal.

Both power and difficulty spring from the same source - *regular expression syntax*. There are entire books on the subject, so please don't feel stupid! It really is pretty heavy going.

You'll want to place the entire regular expression in angle brackets, as in your examples.

The caret (^) *anchors* your regular expression to the start of a line, that is, the part of the URL immediately following the hostname.

Hence <^node> will match http://www.example.com/node, http://www.example.com/node/foo, and http://www.example.com/node/123, if your server is www.example.com.

If you omit this and type <node> for example, your regular expression will match http://www.example.com/node, but also http://www.example.com/nodessert, http://www.example.com/blog/how_to_delete_a_node, etc.

OK so far?

As you have perceived, parentheses *group* sub-expressions (sort of in the same way that parentheses group sub-expressions in arithmetic, or even in written communication, as in this example!). The 'pipe' operator (`|`) is used to separate alternatives. Read it as 'or'. So,

```
<^ (foo) | (bar) >
```

can be read as 'match foo or bar at the start of the line'.

The expression `(?! . . .)` is called a *negative look-ahead operator*. It says to match only if the subexpression following the `?!` does not match from the point where the `?!` is evaluated (matched)

So `<^foo(?! (t))>` matches `http://www.example.com/fool` and `http://www.example.com/foozee` but not `http://www.example.com/foot`, or `http://www.example.com/football`.

One more thing - letters and numbers 'match themselves', but obviously certain other special characters have special meaning (e.g., we've already seen that `^`, `?`, `!`, `|` and parentheses are treated specially). These are called *metacharacters*. To match a metacharacter you need to 'escape' it, by preceding it with a '`\`' (backslash).

So, to match the pattern 'foo?!' you'd have to type `<foo\\?\\!>`.

Now, we can put all of the pieces together. Just as in arithmetic, or in writing compound sentences, you can create long expressions by combining short ones.

Regular expressions can be combined by simply catenating them:`<foo>` matches 'foo', and `<bar>` matches 'bar'; catenate them to get `<foobar>` matching 'foobar'. You can also combine by alternation: `<(foo) | (bar)>` matches foo or bar. You can specify that a subexpressions matches zero or more times by following it with an asterisk: `<(foo)*t>` matches 't', foot, foofoot, foofoofoot, and so on. (I've omitted the server part of the URL for brevity)

On to your example:

```
<^ (?! (node\\/635) | (event) ) >
```

matches (at the beginning of the line) anything that doesn't match the subexpression following `?!` , namely, node/635, or event. I've escaped the slash - I don't know if this is necessary (but it doesn't hurt - any regular character when escaped just matches itself)

If you're beginning to love this stuff, there's hundreds of more pages of fun examples in various books and on other websites (for example at <http://www.php.net>). Just search for 'regular expression' or PCRE (perl compatible regular expressions)

Hope this helps,

Djun

Custom blocks

A custom block contains content supplied by you (as opposed to being generated by a module). Custom content may be either *static* (i.e., HTML) or dynamic (PHP generated content). Virtually all of the functionality of Drupal is accessible from within a PHP content block. The flexibility of blocks provides an extremely powerful way to customize your Drupal website.

You can create a custom block via the block management screen (click on *administer » blocks*). Select the *new* tab and complete the form.

Each custom block has a title, a description, and a body. The content within the body can be as long as you wish.

Note: In Drupal 4.6, the title of the block is used as its identifier in the database table. Therefore you can only have one custom block without a title. This issue (whose status you can track at <http://drupal.org/node/11724>) presents a problem if you don't want certain blocks to have a title.

One workaround for this is to enter a title such as this: <!--title of custom block-->. This satisfies Drupal's need for a block title but the comment markers (<!-- ... -->) prevent it from being displayed.

Examples

This section presents a collection of custom blocks submitted by Drupal users. Other examples of custom block examples are available in the "Blocks" section of the Drupal CVS Repository at
[http://cvs.drupal.org/viewcvs/drupal/contributions/tricks\(blocks/](http://cvs.drupal.org/viewcvs/drupal/contributions/tricks(blocks/).

You can study how the preinstalled Drupal blocks work by searching the associated module files for the string "*_block*". All blocks defined in a module begin with this hook.

Have a look to see what is put into `$blocks["subject"]` and `$blocks["content"]`. The login-block, for example, is implemented in the *user* module's `user_block()` function as block 0 ("case 0:" - this delta is there because the user module defines more than one block).

More examples of PHP code snippets can be found in the snippets section.

A "book-like" navigation block

A recent client site, which I was building with Drupal, needed a more conventional navigation system. Specifically, we needed a 'block' to do for regular pages what the book navigation block does for book.module - list the categories in a taxonomy, and show a sublist of that category when you either click on the category name, or access a page which is inside that category.

Here it is. I've used the "collapsed","expanded" and "leaf" list classes to make it all fit neatly into Drupal's default style.

Caveats: 1) it requires you to apply this long-overdue patch to taxonomy.module and won't work properly without it. 2) It assumes a flat term hierarchy and nodes attached to more than one taxonomy term will be assumed to be a member of only one of them.

Copy and paste this code into a new PHP block, and make a couple of locally-relevant changes (read the comments for details).

```
<?php
switch (arg(0)) {
  case 'node':
    $current = arg(1);
    // Note: Change the '1' below to whatever vocabulary ID
you're using
    $cats   = taxonomy_node_get_terms_by_vocabulary($current,
7);
    foreach ($cats as $cat) {
      $mycategory = $cat->tid;
    }
    break;
  case 'taxonomy':
    $current = 0;
    $mycategory = arg(2);
    break;
  default:
    $current = 0;
    $mycategory = 0;
    break;
}
// Note: Change the '1' below to whatever vocabulary ID
you're using
$terms = taxonomy_get_tree(7);
?>
<div class="menu">
<ul>
<?php
foreach($terms as $term) {
  $looptid = $term->tid;
```

```

$loopname = $term->name;
if($looptid == $mycategory) {
    print "<li class=\"expanded\"><a href=\"$taxonomy/term/$looptid\">$loopname</a><ul>";
    $kiddies =
taxonomy_select_nodes(array($mycategory), 'or', 0, FALSE, 'n.title ASC,
n.sticky DESC');
    if (db_num_rows($kiddies) > 0) {
        while ($getthem = db_fetch_object($kiddies)) {
            print "<li class=\"leaf\"><a href=\"$node/" .
$getthem->nid . "\">$getthem->title</a></li>";
        }
    }
    print "</ul>";
    print "</li>";
} else {
    print "<li class=\"collapsed\"><a href=\"$taxonomy/term/$looptid\">$loopname</a></li>";
}
}
?>
</ul>
</div>

```

Latest stories block

Here is a simple module which displays the titles of the last n changed stories in a block. It was made specifically for my site, and only works on 'story' nodes, though it would be easy to change this. Needs to be topped and tailed with php script open and close angle brackets. [?php // ---- copy from here --- // latest.module v0.1.0, John Clift, 11 Dec 2003 // Module displays a block which lists the titles, linked, // of the last five stories to be added or modified // Database query to get the latest story nodes // \$nlimit sets the number of node titles to display function latest_nodes(\$type) { \$nlimit = 5; \$result = db_query("SELECT n.created, n.title, n.nid, n.changed FROM node n WHERE n.type = '\$type' ORDER BY n.changed DESC LIMIT \$nlimit"); while (\$node = db_fetch_object(\$result)) { \$output .= l(check_output(\$node->title), "node/view/".\$node->nid). "
"; } return \$output; } // Function to display titles of latest story nodes in a block function latest_block(\$op = "list", \$delta = 0) { if (\$op == "list") { \$blocks[0]["info"] = t("Latest n Additions or Changes"); return \$blocks; } else { \$block["subject"] = t("Latest Changes"); \$block["content"] = latest_nodes("story"); return \$block; } } // --- end --- ?]

Show block to certain users only

Note: Some of the examples from this section may not work with versions of Drupal older than 4.5.

Example: you want to show a block only to logged-in users. Site administrators often wish to restrict anonymous users from seeing certain (custom) blocks. Placing the following code in a PHP block will achieve this.

```
<?php
global $user;
if ($user->uid) {
    return "This block is only visible for logged-in users.";
} else {
    return;
}
?>
```

Replace the string "This block is only visible for logged-in users." with anything you want to display only to logged-in users.

The block will not be shown *at all* for users who are not logged-in. You can replace the single `return` in the `else`-block with, e.g.,

```
<?php
return "This block is only visible for logged-in users.";
?>
```

to show some text instead of not displaying the block at all.

Example: Wrapping HTML content so it is visible only to anonymous users. This example uses an if/endif construction to wrap some HTML in a few lines of PHP which will show the content only to anonymous users.

```
<?php
global $user;
if (!$user->uid) : ?>
    <p>This content will be visible only to anonymous
users.</p>
<?php else {
    return;
}
endif;
?>
```

Example: Showing a block only to certain users. If you want to show a certain custom block only to a certain user, you can use the following code in a PHP block:

```
<?php
global $user;
if ($user->uid == 19) {
    return "This block is only visible for the user with the
user-ID 19.";
```

```

} else {
    return;
}
?>
```

Show last 5 nodes by users on their profile pages

This block shows the latest nodes created by users on their profile page.

1. Create a new PHP block as you normally would (*administer » blocks » add block* tab, making sure that you select "PHP code" as the input format). Enter the code below. Change the number '5' to whatever number of posts you want to show.
2. Do not enable the block quite yet. Click *administer » blocks* then the "configure" link next to the block.
3. Scroll down to "Show block on specific pages:", select "Show on only the listed pages." and type in `user/*`. This will only show the block on users' profile pages. (It will only show on numeric pages, such as `user/42` and not `user/register` or `user/logout` because it checks for a numeric user ID.)
4. Click "Save Block"
5. Enable, place and give a weight to the block.

```

<?php
$userid = arg(1);
if (is_numeric($userid)) {
    $nlimit = 5;
    $result = db_query("SELECT n.created, n.title, n.nid,
n.changed
FROM node n
WHERE n.uid = $userid
ORDER BY n.changed
DESC LIMIT $nlimit");
$output .= "<div class=\"item-list\"><ul>\n";
$output .= node_title_list($result);
$output .= "</ul></div>";
return $output;
}
?>
```

Submission queue block

Hi folks, Just in case anyone's interested, I have just written a small custom block which lists items in the submission queue. Create yourself a new PHP block and use this code:

```

uid) { // get the links
$queryResult = db_query_range("SELECT n.*
FROM {node} n WHERE n.moderate = 1", 0, 10);
while ($node = db_fetch_object($queryResult)) {
    if ($user->uid == $node->uid ||
field_get($node->users, $user->uid)) { // it's our own node or we've already voted
```

```
$rows[] = l($node->title, "queue/$node->nid") . (" . queue_score($node->nid) . ");
} else { // it's someone else's node $rows[] = l($node->title, "queue/$node->nid"); } } return
theme("item_list", $rows, "Submission queue") . "<div class=\"more-link\">" . l(t("more"),
"queue", array("title" => t("List all queue entries."))) . "</div>"; } ?>Hope you find this helpful!
Cheers, Mabster
```

Debugging the path

If you want to see the string that is being matched against your pattern add the following code to *block.module*:

```
<?php
print( '<pre> in module: "' . $block['module'] . '" ' .
$block['path'] . '</pre>');
print( '<pre> string: ' . ereg_replace('/(\?q=)?', ' ', request_uri()) . '</pre>');
print( '<pre> matches?: ' . preg_match($block['path'],
ereg_replace('/(\?q=)?', ' ', request_uri())) . '</pre>');
?>
```

It should go into the function *block_list()*, after the following

```
<?php
while ($result && ($block = db_fetch_array($result))) {
    // When the user's account setting is empty, we use the
    // block's regular 'status' (which is the default)
    if ($block['custom'] && $user->uid &&
!isset($user->block[$block['module']][$block['delta']])) {
        $user->block[$block['module']][$block['delta']] =
$block['status'];
        }<br>
**<-- here
?>
```

The output is as follows

```
in module: "menus"
string : sluzby/?q=admin/system/block
matches? : 1
```

Here "matches : 1" means that the regular expression was successful and the block will be displayed. If there is nothing shown for matches, there were no attempt to match "regex"

Custom login

This is a very simple block to display a link to the login page when the user is not logged in and to display a logout link when the user is logged in.

```
<?php
global $user;
if (!$user->uid) {
    // Change the following line's text to whatever you want.
    return '<a href="?q=user/login">Login/Register</a>';
} elseif ($user->uid) {
    // The following line will display the username you are
    // logged in as.
    return 'Logged in as ' . $user->name . '<br><a
href="node?q=logout">Logout</a>';
}
?>
```

If you don't want the block to redundantly show up in the login section, add this to the custom block's path field.

```
<^(!user)>
```

All published content in a list

A very simple block to show all nodes that are published in a list, ordered by creation date.

Its very handy when you have journal that uses not only blog nodes, but also images, weblinks etc.

```
<?php
$result = db_query_range("SELECT n.created, n.title, n.nid,
n.changed
FROM node n
WHERE n.status = 1
ORDER BY n.created
DESC ", 0, 10);
while ($node = db_fetch_object($result)) {
    $output[ ] = l(check_output($node->title),
"node/view/".$node->nid);
}
return theme_item_list($output);
?>
```

Blog categories

Here's a block that shows all of the categories in your Blog vocabulary. It can easily be customized to show the categories from any node type by changing the "vocabulary_node_types.type = 'blog'" part to equal your preferred node type.

```
<?php
  if (user_access('access content')) {
    $result = db_query(db_rewrite_sql("SELECT term_data.tid,
    term_data.name, COUNT(*) AS count FROM {vocabulary_node_types} INNER
    JOIN {term_data} USING (vid) INNER JOIN {term_node} USING (tid) INNER
    JOIN {node} USING (nid) WHERE node.status = 1 AND
    vocabulary_node_types.type = 'blog' GROUP BY term_data.tid,
    term_data.name ORDER BY term_data.name"));
    $items = array();
    while ($category = db_fetch_object($result)) {
      $items[] = l($category->name . ' (' . $category->count
      . ')', 'taxonomy/term/' . $category->tid);
    }
    return theme('item_list', $items);
  }
?>
```

Blogger style 'About Me' block

Although I've only included the avatar, name, location, biography this block could easily be extended to included whatever profile or user information you wish...

```
<?php
  if (arg(0)=='blog'){
    $uid = arg(1);
    $account = user_load(array(is_numeric($uid) ? 'uid' :
    'name') => $uid, 'status' => 1));
    profile_load_profile(&$account);
    echo("<div align=\"left\">");
    if (isset($account->picture)){
      echo(
        '<img style=\"float: left; padding: 2px\"'
        'src=\"$account->picture\" height=\"60px\"'
        '\"');
    }
    if (isset($account->realname)){
      echo(
        '<b>Name:</b><br>'
        '$account->realname<br>'
        '\"');
    }
  }
```

```

if (isset($account->country)) {
  echo("
    <b>Location:</b><br>
    $account->country<br>
    <br>
  ");
}
if (isset($account->biography)) {
  echo("
    <br>
    <em>$account->biography</em><br>
    <br>
  ");
}
echo("
  </div>
  <div align=\"right\">
  <a
  href=\"http://www.blogtown.ca/user/$account->uid\">more</a>
  </div>
");
}
?>

```

Categories block as in 4.5

Create a custom block in PHP format and paste the following to have a categories block like in 4.5:

```

<?php
if (user_access('access content')) {
  $result = db_query("SELECT d.tid, d.name, MAX(n.created) AS
updated, COUNT(*) AS count FROM {term_data} d INNER JOIN {term_node}
USING (tid) INNER JOIN {node} n USING (nid) WHERE n.status = 1 GROUP BY
d.tid, d.name ORDER BY updated DESC, d.name");
  $items = array();
  while ($category = db_fetch_object($result)) {
    $items[] = l($category->name . ' (' . $category->count
. ')', 'taxonomy/term/'. $category->tid) . '<br />'. t('%time
ago', array('%time' => format_interval(time() -
$category->updated)));
  }
  return theme('item_list', $items);
}
?>

```

Blogcentric random image

This is a variation that will display a random image belonging to the current blog's owner...

Set a path of <^blog\/[0-9]> so it only displays on blog pages...

```
<?php
if (arg(0)=='blog'){
  $uid = arg(1);
  $account = user_load(array((is_numeric($uid) ? 'uid' :
  'name') => $uid, 'status' => 1));
  $current_blog = $account->uid;
}
$query = sprintf("SELECT * FROM node WHERE type='image' AND
uid=%d ORDER BY RAND() LIMIT 0,1", $current_blog);
$result = db_query($query);
if (db_num_rows($result) > 0){
  $row = db_fetch_array($result);
  $nid = $row["nid"];
  $title = $row["title"];
  if($title=="")$title="isimsiz";
  $result = mysql_query("SELECT * FROM image WHERE
nid=$nid");
  $row = mysql_fetch_assoc($result);
  $thumb = $row["thumb_path"];
  mysql_free_result($result);
  echo "<div align=\"center\"><a
href=\"http://www.blogtown.ca/node/$nid?res=300x400\"><img
src=\"$thumb\"><br><b>$title</b></a></div>";
}
?>
```

Hits by month

Put this in a block, and it will show the number of hits per month, by month. The number of months shown will depend on what you configured (how long to keep statistics).

The first and last months will be incomplete, because of the log purging.

You can restrict the path of the block to admin to make this visible only to you (the site administrator).

This is MySQL specific. I am sure it can be adapted to other databases.

```
<?php
$header = array ('Month', 'Hits');
$rows = array();
```

```

$q = "SELECT DATE_FORMAT(FROM_UNIXTIME(timestamp), '%Y-%m')
AS month, COUNT(*) AS hits FROM {accesslog} GROUP BY month ORDER BY
month DESC";
$result = db_query ($q);
while ( $row = db_fetch_object ( $result ) ) {
  $rows[] = array ( 'data' => array ( $row->month, $row->hits
)) ;
}
if (! $rows) {
  $rows[] = array(array('data' => t('No log data
available.'), 'colspan' => 2));
}
print theme('table', $header, $rows);
?>

```

One variation that can cut off the hits to the month is to replace the SQL above by:

```

<?php
SELECT DATE_FORMAT(FROM_UNIXTIME(timestamp), '%Y-%m') AS
month, COUNT(*) AS hits FROM accesslog
WHERE FROM_UNIXTIME(timestamp) >
DATE_SUB(FROM_UNIXTIME(UNIX_TIMESTAMP()), INTERVAL 3 MONTH)
GROUP BY month ORDER BY month DESC
?>

```

Comment approval count block

Displays a count of the comments waiting to be approved to the super-admin user, as well as a link to the comment approval queue.

```

<?php
global $user;
if ($user->uid == 1) {
  $sql = 'SELECT c.nid, c.cid, c.timestamp, c.status, c.name,
c.homepage FROM {comments} c WHERE c.status = 1';
  $result = db_query($sql);
  $num_comments = db_num_rows($result);
  return "<div align='center'>
    <p>There are <strong>$num_comments</strong>
comment(s) waiting to be
    <a
      href='admin/comment/list/approval'>approved</a></p></div>";
}
?>

```

quicker way might be this:

```
<?php
$comment_count = db_result(db_query("SELECT count(*) FROM
comments WHERE status=1"));
?>
```

Counter (x days before / past...)

A small block that displays the days after a certain date.

```
<?php
$day = 30;
$month = 12;
$year = 1969;
$age= ((int)((mktime (0,0,0,$month,$day,$year) -
time(void))/86400) * -1 );
print ("My age in days is " . $age . ".");
?>
```

Paypal blocks

Here are two snippets that display how much and who has made donations. Requires the paypal_framework module.

Total money sent to you:

```
<?php
$result = db_query(
    "SELECT SUM(payment_gross) as total FROM paypal_log
");
$ppt = db_fetch_object($result);
print sprintf( "%01.2f", $ppt->total+0 );
?>
```

The 10 most recent people to have sent you the money:

```
<?php
$result = db_query(
    "SELECT business,option_name1 FROM paypal_log ORDER BY
payment_date DESC LIMIT 10
");
$count = 0;
while( $donor = db_fetch_object($result) ) {
    print "<li>";
    if( $donor->{business} ){
        print $donor->{business};
    }elseif( $donor->{option_name1} ){
        print $donor->{option_name1};
    }
?>
```

```

        print $donor->{option_name1};
    }else{
        print "<i>anonymous</i>";
    }
    print "</li>";
    $count = $count + 1;
}
if( $count==0 ){
    print "<li><i>none</i></li>";
}
?>

```

Show highest contributers to a site

From a post by javanaut in answer to a forum question. This works in Drupal 4.5

```

<?php
$users = db_query("SELECT COUNT(cid) AS count, {users}.uid,
{users}.name FROM {comments} LEFT JOIN {users} ON {comments}.uid =
{users}.uid WHERE {comments}.uid != 0 GROUP BY uid ORDER BY count DESC
LIMIT 20");
print "<ul>";
while ($user = db_fetch_object($users)) {
    print "<li>".l($user->name,"user/$user->uid")."
($user->count)</li>";
}
print "</ul>";
?>

```

Top users by comment number

This block displays the top 10 users who made the most comments, in descending order.

Note: to create a similar block that counts the users who authored the most nodes (pages/stories/forum articles, ...etc.) change the word "comments" with "node" in the select statement below.

```

<?php
$header = array ('User', 'Posts');
$rows = array();
$q = "SELECT u.name, count(*) AS posts FROM {users} u,
{comments} c WHERE u.uid = c.uid GROUP by u.name ORDER BY posts DESC
LIMIT 10";
$result = db_query ($q);
while ( $row = db_fetch_object ( $result ) ) {
    $rows[ ] = array ( 'data' => array ( $row->name, $row->posts
)) ;

```

```

}
if (!$rows) {
  $rows[] = array(array('data' => t('No comments
available.'), 'colspan' => 2));
}
print theme('table', $header, $rows);
?>

```

Uptime and load on Unix systems

This block makes it possible to show the current load and uptime in a block. The code will not work as is on all Unixes. One might consider it a security risk to show what your uptime is (*kernel patching requires reboot*), so .. batteries not included, always look both sides when crossing a road etc.

You can learn from this code how to execute shell commands from php.

```

<?php
$uptime = shell_exec("cut -d. -f1 /proc/uptime");
$loadavg_array = explode(" ", exec("cat /proc/loadavg"));
$loadavg = $loadavg_array[2];
$days = floor($uptime/60/60/24);
$hours = $uptime/60/60%24;
$mins = $uptime/60%60;
$secs = $uptime%60;
echo "This server has been up $days day(s) $hours hour(s)
$mins minute(s) and $secs second(s)";
echo "<p><br>";
print("[ Current server load: " . $loadavg . " ]");
?>

```

Pulldown top level category links

This is a little block to show a pull-down form list of your top level taxonomies. When you select a list item a little Javascript will jump you to that taxonomy's page.

```

<?php
//Default Value for Select
$t_name[] = 'Jump to Category';
//Get Taxonomy Terms with No Parent
$result = db_query("SELECT term_data.name, term_data.tid FROM
term_data, term_hierarchy WHERE term_data.tid=term_hierarchy.tid AND
term_hierarchy.parent = 0");
//Populate array with names and links.  The key of the array
will be our URL.
while ($term = db_fetch_object($result)) {
  $t_name['taxonomy/term/' . $term->tid] = $term->name;
}

```

```

}
//Build the select list using Drupal's 'form_select'
function.
$category_select = form_select('', 'category', '', $t_name,
$description = NULL,
'onChange="top.location.href=document.getElementById(\'edit-category\').options[document.getElementById(\'edit-category\').selectedIndex].value"',
$multiple = FALSE, $required = FALSE);
//Give our subsequent form a name we can grab with the
Javascript
$form_attributes[name] = 'form';
//Build our form with Drupal's 'form' function.
return form($category_select, $method = 'get', '',
$form_attributes);
?>

```

This isn't really such a great thing to do as far as accessibility though. If you are using any device other than a mouse it is hard to select the options, with the javascript in there auto directing your page location.

So here is a version of the same script with a submit button.

```

<?php
//Default Value for Select
$t_name[] = 'Jump to Category';
//Get Taxonomy Terms with No Parent
$result = db_query("SELECT term_data.name, term_data.tid FROM
term_data, term_hierarchy WHERE term_data.tid=term_hierarchy.tid AND
term_hierarchy.parent = 0");
//Populate array with names and links. The key of the array
will be our URL.
while ($term = db_fetch_object($result)) {
  $t_name['taxonomy/term/' . $term->tid] = $term->name;
}
$category_select = form_select('', 'category', '', $t_name,
$description = NULL, '', $multiple = FALSE, $required =
FALSE);
//Add button javascript
$button_attributes[onClick] =
'top.location.href=document.getElementById(\'edit-category\').options[document.getElementById(\'edit-category\').selectedIndex].value';
//Add button
$submit_button = form_button('Go To Category!', $name = 'op',
$type = 'button', $button_attributes);
//Create var for all form contents
$form_info = $category_select . $submit_button;
//Give our subsequent form a name we can grab with the
Javascript
$form_attributes[name] = 'form';
//Build our form with Drupal's 'form' function.

```

```
return form($form_info, $method = 'get', '',
$form_attributes);
?>
```

Drupal modules and features

The nodes below contain the help available for the Drupal modules.

Aggregator: syndicating content

The news aggregator is a powerful on-site RSS syndicator/news reader that can gather fresh content from news sites and weblogs around the web.

Users can view the latest news chronologically in the main news aggregator display or by source. Administrators can add, edit and delete feeds and choose how often to check for newly updated news for each individual feed. Administrators can also tag individual feeds with categories, offering selective grouping of some feeds into separate displays. Listings of the latest news for individual sources or categorized sources can be enabled as blocks for display in the sidebar through the block administration page. The news aggregator requires cron to check for the latest news from the sites to which you have subscribed. Drupal also provides a machine-readable OPML file of all of your subscribed feeds.

You can

- administer your list of news feeds [administer >> aggregator](#).
- add a new feed [administer >> aggregator >> add feed](#).
- add a new category [administer >> aggregator >> add category](#).
- configure global settings for the news aggregator [administer >> settings >> aggregator](#).
- control access to the aggregator module through access permissions [administer >> access control >> permissions](#).

Old page

Thousands of web sites, especially news sites and weblogs, syndicate their most recent site content for others to display. The syndicated content always includes titles, also known as headlines, for the newest published stories. Each headline acts as a direct link to the stories on the remote site. Along with the headline, most sites typically provide either the first few paragraphs of the story or a short summary. Many individuals use client-based news aggregators on their personal computer to aggregate content, such as FeedDemon (for Windows), NetNewsWire (for Macs) and Amphetadesk (Windows, Mac and Linux).

Drupal also has a news aggregator built in as a standard feature. With it, you can subscribe to feeds from other sites and display their content for your site users. Simply enable the aggregator module in administer Â» modules, then click administer Â» aggregator and enter the feeds that you choose.

What do I need to subscribe to a feed?

The standard method of syndication is using the XML-based RSS format. RSS stands for Really Simple Syndication, RDF Site Summary, or Rich Site Summary, depending on whom you talk to. To syndicate a site's content, obtain the full URL of the RSS page providing syndication. Common file tags for RSS pages are .rss, .xml and .rdf. Example: <http://slashdot.org/slashdot.rdf>.

Most weblog sites that offer syndication will have an obvious link on the main page. Often you need only look for a red XML button, such as the one Drupal uses for site syndication.

Some sites do not make their RSS feeds as easy to find. Or maybe you want to find a number of feeds on a given topic, without extensively searching the web. In that case, try an RSS syndication directory such as Syndic8.

To learn much more about RSS, here are some good introductions:

- Mark Pilgrim's What is RSS
- WebReference.com's The Evolution of RSS

NOTE: Enable your site's XML syndication button by turning on the Syndicate block in administer Â» blocks.

Configuring news feeds

To subscribe to an RSS feed on another site, click *administer Â» aggregator*.

Once there, select the *add feed* tab at the top of the aggregator administration page. Drupal will then ask for the following:

- **Title** -- The text entered here will be used in your news aggregator, within the administration configuration section, and as title for the news feed block. As a general rule, use the web site name from which the feed originates.
- **URL** -- Here you'll enter the fully-qualified URL for the feed for the site you want to subscribe to.
- **Update interval** -- The update interval is how often Drupal will automatically access the RSS URL for the site for fresh content. The 1 hour default is typically a good minimum to use. Accessing another site's RSS page more frequently can be considered impolite because it requires the other site's server to handle your automatic requests. To take advantage of this feature, note that cron.php must be configured to have your feeds updated regularly. Otherwise, you'll have to manually update feeds one at a time within the

news aggregation administration (*administer » aggregator*) section.

Once you submit your new feed, check to see if it is working properly. Select *update items* on the main news aggregation page. If you do not see any items listed for that feed, edit the feed and make sure that the URL was entered correctly.

Creating categories in the aggregator

1. Go to *administer » aggregator* then click on the add category tab.
2. Add a title to the category, then a description.
3. If you wish to have a block of the last x items from that category, select the number of items in "Latest items block". To place the block on your sidebar, go to *administer » blocks* and look for the category you just created.

Now every time you add a feed, you can select a category which the items will automatically appear under. Alternatively, you can tag individual items in your aggregator to appear in a category.

Tagging individual items in the aggregator

1. To get to the categorization screen, in your sidebar navigation, click *news aggregator*
2. Here you have two options:
 1. click *categories*, then the category you wish to look at, then the categorize tab.
 2. click *sources*, then the feed source you wish to look at, then the categorize tab.
3. You will be presented with a list of items to categorize, plus to the right, the categories which you can assign to each item. If you have the multiple-select option enabled in the aggregator configuration, you can select more than one category for an item by holding down CTRL (PC) or CMD (Mac) and clicking on each category.

Using the news aggregator

The news aggregator has a number of ways that it displays your subscribed content:

- **Latest News** -- Displays all incoming content in the order received with
 - The title of the original post.
 - The name of the source, which acts as a link to an individual feed page, listing information about that feed and incoming content for that feed only.
 - A description, the first few paragraphs or summary of the originating post (if any).
 - The list of categories that the feed (or feed item) belongs to, with a link to

each category.

- **News by Source:** Organizes incoming content by feed, displaying titles which link to the originating post. Also has an icon which acts as blog it link.
- **News by Topic:** Organizes incoming content by bundles, displaying titles which link to the originating post. Also has an icon which acts as blog it link.
- **News Sources:** Displays an alphabetical listing of all subscribed feeds and a description. The title acts as a link to an individual feed page, listing information about that feed and incoming content for that feed only.

RSS feed blocks

In addition to providing subscribed content through the news aggregator, Drupal automatically can create a block for every feed as well as every category, though the administrator can choose whether or not a feed or category gets its own blocks by configuring the individual feeds and categories. Enable any or all of the blocks using block management by clicking *administer >> blocks*.

Archive: view content by date

The archive page allows content to be viewed by date. It also provides a monthly calendar view that users can use to navigate through content.

To view the archive by date, select the date in the calendar. Administrators can enable the *browse archives* block in block administration to allow users to browse by calendar. Clicking on a date in the monthly calendar view shows the content for that date. Users can navigate to different months using arrows beside the month's name in the calendar display. The current date will be highlighted in the calendar.

You can

- view your archive by day.
- enable the *browse archives* block at *administer >> block*.

Block: controlling content in the sidebars

Blocks are the boxes of related/grouped data that are visible in the sidebar(s) of your web site. These are usually generated automatically by modules (e.g. recent forum topics), but administrators can also create their own defined blocks.

The sidebar each block appears in depends on both which theme you are using (some are left-only, some right, some both), and on the settings in block management.

The block management screen lets you specify the vertical sort-order of the blocks within a sidebar. You do this by assigning a weight to each block. Lighter blocks (smaller weight) "float up" towards the top of the sidebar. Heavier ones "sink down" towards the bottom of it.

A block's visibility depends on:

- Its enabled checkbox. Disabled blocks are never shown.
- Its throttle checkbox. Throttled blocks are hidden during high server loads.
- Its path options. Blocks can be configured to only show/hide on certain pages.
- User settings. Administrators can choose to let your users decide whether to show/hide certain blocks.
- Its function. Dynamic blocks (such as those defined by modules) may be empty on certain pages and will not be shown.

Module blocks

Module blocks are available when modules are enabled. These blocks can be administered in block administration.

Administrator defined blocks

An administrator defined block contains content supplied by the administrator. Each admin-defined block consists of a title, a description, and a body which can be as long as you wish. The Drupal engine will render the content of the block.

You can

- enable throttle and configure blocks at administer >> block.
- add a block at administer >> block >> add block.

Blog: a blog for every user

The blog module allows registered users to maintain an online weblog (commonly known as a blog), often referred to as an online journal or diary. Blogs are made up of individual posts that are time stamped and are typically viewed by date as you would a diary. Blogs often contain links to webpages users have read and/or agree/disagree with.

The blog module adds a *user blogs* navigation link to the site, which takes any visitor to a page that displays the most recent blog entries from all the users on the site. The navigation menu has a *create a blog entry* link (which takes you to a submission form) and a *view personal blog* link (which displays your blog entries as other people will see them). The blog module also creates a *recent blog posts* block that can be enabled.

If a user has the ability to post blogs, then the import module (news aggregator) will display a blog-it link next to each news item in its lists. Clicking on this takes the user to the blog submission form, with the title, a link to the item, and a link to the source into the body text already in the text box, ready for the user to add a comment or explanation. This actively encourages people to add blog entries about things they see and hear elsewhere in the website and from your syndicated

partner sites.

You can

- read your blog via your user profile at my account.
- post a blog at create content >> personal blog entry.
- administer blog at administer >> content >> configure >> content types >> personal blog entry.
- administer blog api at administer >> settings >> blogapi.
- enable the "recent blog posts" block at administer >> blocks.

HOWTO: Configure user blogs

To implement user blogs on your Drupal site, turn on the blog module and enable permissions:

1. Go to *administer* Â» *modules* and check the box in the status column to the right of *blog*.
2. Under *administer* Â» *access control*, check the *edit own blog* box for each role you wish to maintain blogs.

Aftewards, once logged in, each user with the permission to maintain a blog will be able to click *create content* Â» *personal blog entry* and will see *my blog* (which displays blog entries as other people will see them) in the user navigation block. At the top of each individual blog post, the original blog author will find an *edit* tab.

To add instructions for users on creating their blogs and set workflow options such as *published*, *promoted to the front page*, etc.:

1. Select *administer* Â» *content* Â» *configure* Â» *content type*, then the "configure" link next to "personal blog entry"
2. Enter your instructions in the available text field.
3. Set workflow options.
4. Use the *Minimum number of words in a blog entry* setting to specify a minimum length for all blog posts.

What is a blog or weblog?

Drupal's blog module allows all registered users to maintain a personal weblog on site. Blogs are easily- and frequently-updated websites usually written in an informal and conversational style. They are ordered reverse-chronologically (that is, the most recent entry is at the top) and have archives of past entries. Each individual entry has a permanentâthat is to say, stableâURL linking directly to that item. Blogs typically have comments for each entry so that readers can participate in the discussion, and they usually have RSS feed to be syndicated elsewhere or read in an desktop aggregator. Each entry usually contains one idea, with a link to the source of the original item being discussed. Blogs can be (and are) written

about any subject, from daily personal life to technology to politics to knitting to sports to a company's products.

From a more practical standpoint, blogs can be seen as a means of personal knowledge publishing, a place for researchers or enthusiasts to build and share knowledge about their interests. Or in project oriented sites, as a workspace for project members to post ideas for commenting by others in a group.

For a more complete definition of blogging with links to resources and examples, see George Siemens' [The Art of Blogging - Part 1](#) and [The Art of Blogging - Part 2](#).

Making user blogs more accessible

Drupal provides a number of ways to make user blog posts accessible. You'll need to decide which ones work best for how your Drupal site is configured:

- **A link in the navigation bar:** After activating the blog module, most Drupal themes will include a *Blogs* link in the header navigation bar. The user blog listing contains the most recent blog posts by all site users. If the site is using xtemplate, you'll need to create the link yourself. Go to *site configuration* → *themes* → *xtemplate* and add in the HTML to create the URL (to find the URL, switch your site theme momentarily to *Marvin* and the link will be present in the navigation header).
- **Making user blog listings the default home page:** Click *administer* → *settings* and type in the word "blog" (without quotes) for *Default front page*.
- **Promoting individual blog posts:** If "node" is the *Default front page* setting, administrators can elect to promote any user blog posts to the front page. Click onto the blog post you want to promote, then the *edit* tab and then check *Promoted to Front Page*.
- **Promoting individual blog posts automatically:** Click *administer* → *content* → *configure* → *content type*, then the "configure" link next to "personal blog entry", and check the *Promoted front page* box in the "Default options" group. (This will work if "node" is the *Default front page* in *administer* → *settings*.)
- **Links to recently-updated blogs on the sidebar:** Drupal also makes available a Most recent blogs block under *administer* → *blocks*.

Additional features

- **Blog it:** Users with blogs will see a "blog it" link in the form of a linked image i.e. when viewing posts in the news aggregator. Other news listings, such as RSS blocks, will have an icon in place of the textual blog it link. When the blog it option is selected, the user will be taken to the blog entry form, with the title, a link to the item, and a link to the source already entered in the text input field, ready for the user to add explanation.
- **User Blog RSS syndication:** each individual user blog has their own RSS feed, allowing other sites to syndicate their content or allowing readers to read the individual blog in an aggregator. To find the RSS feed for a user, view their

personal blog (in their personal information, which you can get to by clicking on their username, select *view recent blog entries*). Then look for the XML icon at the bottom of their blog page.

BlogApi: post from blog tools

The blog API module enables a post to be posted to a site via external GUI applications. Many users prefer to use external tools to improve their ability to read and post responses in a customized way. The blog api provides users the freedom to use the blogging tools they want but still have the blogging server of choice.

When this module is enabled and configured you can use programs like Ecto to create and publish posts from your desktop. Blog API module supports several XML-RPC based blogging APIs such as the Blogger API, MetaWeblog API, and most of the Movable Type API. Any desktop blogging tools or other services (e.g. Flickr's "post to blog") that support these APIs should work with this site.

This module also allows site administrators to configure which content types can be posted via the external applications. So, for instance, users can post forum topics as well as blog posts. Where supported, the external applications will display each content type as a separate "blog".

You can

- view the XML-RPC page on your site at >> /xmlrpc.php.
- administer >> settings >> blog api.

Book: collaborative document publishing

The *book* content type is suited for creating structured, multi-page hypertexts such as site resource guides, manuals, and Frequently Asked Questions (FAQs). It permits a document to have chapters, sections, subsections, etc. Authors with suitable permissions can add pages to a collaborative book, placing them into the existing document by adding them to a table of contents menu.

Books have additional *previous*, *up*, and *next* navigation elements at the bottom of each page for moving through the text. Additional navigation may be provided by enabling the *book navigation block* on the block administration page.

Users can select the *printer-friendly version* link visible at the bottom of a book page to generate a printer-friendly display of the page and all of its subsections. They can choose to *export* the page and its subsections as DocBook XML (for offline editing, or production of print or other electronic publication formats), or as an outline (titles only), by selecting the *export DocBook XML* and *export OPML* links respectively. DocBook export currently treats node content as preformatted text.

Administrators can view a book outline, from which it is possible to change the titles of sections, and their *weight* (thus reordering sections). From this outline, it is also possible to edit and/or delete book pages. Many content types besides pages (for example, blog entries, stories, and polls) can be added to a collaborative book by choosing the *outline* tab when viewing the post.

You can

- create new book pages: create content >> book page
- administer individual books (choose a book from list): administer >> content >> books
- set workflow and other global book settings on the book configuration page: administer >> content >> configure >> content types >> book page
- enable the book navigation block: administer >> block
- control who can create, edit, and maintain book pages by setting access permissions: administer >> access control

Maintaining a FAQ using a collaborative book

Collaborative books let you easily set up a Frequently Asked Questions (FAQ) section on your web site. The main benefit is that you don't have to write all the questions/answers by yourself - let the community do it for you!

In order to set up the FAQ, you have to create a new book which will hold all your content. To do so,

1. Click on the create content >> book page link.
2. Give it a thoughtful title. A title like "Estonia Travel - FAQ" is nice.
3. Set the *Parent* to *<top-level>* to make this page the beginning of a new book
4. Add in the text of the book page into the *Body* textarea.
5. Leave the *log message* blank for now.
6. Use the *weight* field to position this book in the list on the book module page.

After you have submitted this book page, you are ready to begin filling up your book with questions that are frequently asked.

Creating new pages for your FAQ. The process for creating new pages in your FAQ is very similar to above. When choosing the *Parent* option, select the FAQ book page that you already created to add it to the book.

Adding existing non-book pages to your FAQ. Whenever you come across a post which you want to include in your FAQ,

1. Click on the *outline* tab at the top of the page.
2. Place the relevant post wherever is most appropriate in your book by selecting a *Parent*.

Notes:

- Books are quite flexible. They can have sections like *Flying to Estonia, Eating in Estonia* and so on. As you get more experienced with the book module, you can reorganize posts in your book so that it stays organized.
- Any comments attached to those relevant posts which you designate as book pages will also be transported into your book. This is a great feature, since much wisdom is shared via comments. Remember that all future comments and edits will automatically be reflected in your book.
- You may wish to edit the title of posts when adding them to your FAQ. Clear titles improve navigability enormously.
- Book pages may come from any content type (blog, story, page, etc.). If you are creating a post solely for inclusion in your book, then use the create content → book page link.
- If you don't see the *edit* link or *outline* tab, then you probably have insufficient permissions.

Comment: allow comments on content

The comment module creates a discussion board for each post. Users can post comments to discuss a forum topic, weblog post, story, collaborative book page, etc. The ability to comment is an important part of involving members in a community dialogue.

An administrator can give comment permissions to user groups, and users can (optionally) edit their last comment, assuming no others have been posted since. Attached to each comment board is a control panel for customizing the way that comments are displayed. Users can control the chronological ordering of posts (newest or oldest first) and the number of posts to display on each page. Comments behave like other user submissions. Filters, smileys and HTML that work in nodes will also work with comments. The comment module provides specific features to inform site members when new comments have been posted. On sites with active commenting from users, the administrator can turn over comment moderation to the community.

You can

- control access for various comment module functions through access permissions administer >> access control.
- administer comments administer >> comments >> configure.

Old page

When enabled, the Drupal comment module creates a discussion board for each Drupal node. Users can post comments to discuss a forum topic, weblog post, story, collaborative book page, etc. An administrator can give comment permissions to user groups, and users can (optionally) edit their last comment,

assuming no others have been posted since.

User control of comment display

Attached to each comment board is a control panel for customizing the way that comments are displayed. Users can control the chronological ordering of posts (newest or oldest first) and the number of posts to display on each page. Additional settings include:

- **Threaded** â Displays the posts grouped according to conversations and subconversations.
- **Flat** â Displays the posts in chronological order, with no threading whatsoever.
- **Expanded** â Displays the title and text for each post.
- **Collapsed** â Displays only the title for each post.

When a user chooses *save settings*, the comments are then redisplayed using the user's new choices. Administrators can set the default settings for the comment control panel, along with other comment defaults, in administer Â» comments Â» configure. NOTE: When comment moderation is enabled, users will have another control panel option to control thresholds (see below).

Additional comment configurations

Comments behave like other user submissions in Drupal. Filters, smileys and HTML that work in nodes will also work with comments. Administrators can control access to various comment module functions through administer Â» access control Â» permissions. Know that in a new Drupal installation, all comment permissions are disabled by default. The choice of which permissions to grant to which roles (groups of users) is left up to the site administrator. The following permissions:

- **Access comments** â Allows users to view comments.
- **Administrate comments** â Allows users complete control over configuring, editing and deleting all comments.
- **Moderate comments** â Allows users to rate comment postings (see more on moderation below).
- **Post comments** â Allows users to post comments into an administrator moderation queue.
- **Post comments without approval** â Allows users to directly post comments, bypassing the moderation queue.

Notification of new comments

Drupal provides specific features to inform site members when new comments have been posted.

Drupal displays the total number of comments attached to each node, and tracks comments read by individual site members. Members which have logged in will see a notice accompanying nodes which contain comments they have not read. Some administrators may want to download, install and configure the notify module. Users can then request that Drupal send them an e-mail when new comments are posted (the notify module requires that cron.php be configured properly).

The *tracker* module, disabled by default, displays all the site's recent posts. There is a link to the recent posts page in the navigation block. This page is a useful way to browse new or updated nodes and comments. Content which the user has not yet read is tagged with a red star (this graphic depends on the current theme). Visit the comment board for any node, and Drupal will display a red "*new*" label beside the text of unread comments.

Comment moderation

On sites with active commenting from users, the administrator can turn over comment moderation to the community.

With comment moderation, each comment is automatically assigned an initial rating. As users read comments, they can apply a vote which affects the comment rating. At the same time, users have an additional option in the control panel which allows them to set a threshold for the comments they wish to view. Those comments with ratings lower than the set threshold will not be shown. To enable moderation, the administrator must grant moderate comments permissions. Then, a number of options in administer Â» comments Â» configure must be configured.

Moderation votes

The first step is to create moderation labels which allow users to rate a comment. Go to administer Â» comments Â» configure Â» moderation votes. In the *vote* field, enter the textual labels which users will see when casting their votes. Some examples are

- Excellent +3
- Insightful +2
- Useful +1
- Redundant -1
- Flame -3

So that users know how their votes affect the comment, these examples include the vote value as part of the label, although that is optional. Using the weight option, you can control the order in which the votes appear to users. Setting the weight heavier (positive numbers) will make the vote label appear at the bottom of the list. Lighter (a negative number) will push it to the top. To encourage positive voting, a useful order might be higher values, positive votes, at the top, with negative votes at the bottom.

Moderator vote/values matrix

Next go to administer » comments » configure » moderation matrix. Enter the values for the vote labels for each permission role in the vote matrix. The values entered here will be used to create the rating for each comment. NOTE: Comment ratings are calculated by averaging user votes with the initial rating.

Creating comment thresholds

In administer » comments » configure » moderation thresholds, you'll have to create some comment thresholds to make the comment rating system useful. When comment moderation is enabled and the thresholds are created, users will find another comment control panel option for selecting their thresholds. They'll use the thresholds you enter here to filter out comments with low ratings. Consequently, you'll probably want to create more than one threshold to give users some flexibility in filtering comments.

When creating the thresholds, note that the *Minimum score* is asking you for the lowest rating that a comment can have in order to be displayed. To see a common example of how thresholds work, you might visit Slashdot and view one of their comment boards associated with a story. You can reset the thresholds in their comment control panel.

Initial comment scores

Finally, you may want to enter some *initial comment scores*. In administer » comments » configure » moderation roles you can assign a beginning rating for all comments posted by a particular permission role. If you do not assign any initial scores, Drupal will assign a rating of **0** as the default.

Contact: allow other users to contact you

The contact module allows other users to contact you by e-mail via your personal contact form. Users can send a subject and message in the contact form. The contact module is important in helping make connections among members of your community.

Users can administer the contact settings in their account settings. Note that a users e-mail address is not made public and that privileged users such as site administrators are able to contact you even if you choose not to enable this feature. If users activate the personal contact form, then a contact tab will appear in their user profile.

You can

- view user profiles.
- enable the personal contact form in administer >> user >> edit tab >> account settings tab >> personal contact settings.

Drupal: Drupal sites directory server

The Drupal module uses the XML-RPC network communication protocol to connect your site with a directory server that maintains a directory of sites. Community leaders who have common interests may wish to be part of a larger community and showing sites in a common directory is a good way to do this.

Enabling this module will:

- list your site in a site directory.
- allow members on all sites using the Drupal module to login to your site without registering using their distributed identification and vice versa.
- allow members to login to any other site which uses the Drupal module, using a login name which looks much like an email address for your site: *username@example.com*

The Drupal module administration page allows you to set the xml-rpc server page. The listing of your site in a site directory will occur shortly after your sites next cron run.

You can

- run your cron job at your sites cron page.
- browse to the XML-RPC site directory.
- view your XML-RPC page.
- administer Drupal administer >> settings >> drupal.

Old page

The "Drupal" module features a capability whereby other drupal sites may *call home* to report their existence. In turn, this enables a pod of Drupal sites to find, cooperate and advertise each other.

Currently, the main application of this feature is the Drupal sites page. By default, fresh Drupal installations can use drupal.org as their *directory server* and report their existence. This reporting occurs via scheduled XML-RPC pings.

Drupal administrators should simply enable this feature to get listed on the Drupal sites page. Just set your site's name, e-mail address, slogan and mission statement on the administer » settings page. Then make sure that the field called *Drupal XML-RPC server* on the administer » settings » drupal page is set to <http://www.drupal.org/xmlrpc.php>, and enable this feature using the dropdown directly below.

The listing of your site will occur shortly after your site's next cron run. Note that cron.php should be called using the domain name which you want to have listed at drupal.org. For example, don't kick off cron by requesting <http://127.0.0.1/cron.php>. Instead, use a publicly accessible domain name such as

<http://www.example.com/cron.php>.

Also note that your installation need not use drupal.org as its directory server. For example, this feature is perfectly capable of aggregating pings from all of your departmental drupal installations sites within an enterprise.

Filter: Input formats for user content

The filter module allows administrators to configure text input formats for the site. For example, an administrator may want a filter to strip out malicious HTML from user's comments. Administrators may also want to make URLs linkable even if they are only entered in an unlinked format.

Users can choose between the available input formats when creating or editing content. Administrators can configure which input formats are available to which user roles, as well as choose a default input format. Administrators can also create new input formats. Each input format can be configured and have its position rearranged.

You can

- select the default filter and which roles can use filters at administer >> filter.
- configure each filter at administer >> filter >> configure

Forum: create threaded discussions

The forum module lets you create threaded discussion forums for a particular topic on your site. This is similar to a message board system such as phpBB. Forums are very useful because they allow community members to discuss topics with one another, and they are archived for future reference.

Forums can be organized under what are called *containers*. Containers hold forums and, in turn, forums hold threaded discussions. Both containers and forums can be placed inside other containers and forums. By planning the structure of your containers and forums well, you make it easier for users to find a topic area of interest to them.

You can

- administer forums at administer >> forums.

HOWTO: Create a forum

Users can post topics to forums. Forums can appear below other forums in a hierarchy, or can appear in containers.

1. Click administer Â» forums.
2. Click the *add forum* tab.
3. Type in a forum name.

4. Type in a forum description. This will show up below the forum name when users visit the forum section.
5. Select a parent.
 - You can choose either a forum or a container to be a 'parent' for a forum. If there are no containers or existing forums to choose from, you will only see <root> as the option, meaning it will be at the top of the forum hierarchy.
6. Select a weight. Higher numbers means the container will sink down in the list, lower numbers mean the container will rise up.
7. Click the "Submit" button.

HOWTO: Create forum containers

Containers are 'parent' for other forums, that is, users cannot post topics to containers but they hold forum forums to which users can post.

1. Click *administer Â» forums*.
2. Click the *add container* tab.
3. Type in a container name.
4. Type in a container description. This will show up below the container name when users visit the forum section.
5. Select a parent. If there are no containers or existing forums to choose from, you will only see <root> as the option, meaning it will be at the top of the forum hierarchy.
6. Select a weight. Higher numbers means the container will sink down in the list, lower numbers mean the container will rise up.
7. Click the "Submit" button.

Help: context sensitive information

The help module displays context sensitive help information. Users can learn how to use modules and accomplish tasks quicker with less errors by clicking on links in provided by the help module.

Modules can make documentation available to other modules with this module. All user help should be presented using this module. Some examples of help:

- The name of a module (unused, but there)
- The description found on the admin/system/modules page.
- The module's help text, displayed on the admin/help page and through the module's individual help link.
- The help for a distributed authorization module (if applicable).
- The description of a post type (if applicable).

You can

- not administer the help system.

Legacy: remapping of old-style URLs

The legacy module provides legacy handlers for upgrades from older installations. These handlers help automatically redirect references to pages from old installations and prevent *page not found* errors for your site.

The legacy module handles legacy style taxonomy page, taxonomy feed, and blog feed paths. It also handles URL upgrades from Drupal 4.1. It rewrites old-style URLs to new-style URLs (clean URLs).

Example Mappings:

- *taxonomy/page/or/52,97* to *taxonomy/term/52+97*.
- *taxonomy/feed/or/52,97* to *taxonomy/term/52+97/0/feed*.
- *blog/feed/52* to *blog/52/feed*.
- *node/view/52* to *node/52*.
- *book/view/52* to *node/52*.
- *user/view/52* to *user/52*.

Legacy module has no configurable options.

Locale: multi-language support

The locale module allows you to present your Drupal site in a language other than the default English. You can use it to set up a multi-lingual web site or replace given *built-in* text with text which has been customized for your site. Whenever the locale module encounters text which needs to be displayed, it tries to translate it into the currently selected language. If a translation is not available, then the string is remembered, so you can look up untranslated strings easily.

The locale module provides two options for providing translations. The first is the integrated web interface, via which you can search for untranslated strings, and specify their translations. An easier and less time-consuming method is to import existing translations for your language. These translations are available as *GNU gettext Portable Object files* (*.po* files for short). Translations for many languages are available for download from <http://drupal.org>.

If an existing translation does not meet your needs, the *.po* files are easily edited with special editing tools. The locale module's import feature allows you to add strings from such files into your site's database. The export functionality enables you to share your translations with others, generating Portable Object files from your site strings.

You can

- administer localization at administer >> localization.
- manage strings for the localization: administer >> localization >> manage strings.
- add a locale language: administer >> localization >> add language .

HOWTO: Creating a customized language set to replace Drupal terminology

1. Enable the locale module on the administer » modules page
2. Go to the administer » localization page.
3. Select the **add language** tab .
4. Assuming English, create a custom language by adding **en-US** in the Language code text field.
5. Give your language a name, such as **custom-English** (be sure not to use spaces in your language name), and add the language.
6. This will return you to the main localization page. Set your new language as enabled and as the default.
7. Save the configuration.
8. Then disable the original **English** language set (that is, unless you would like users to be given the option to choose between the two in their account area).

Now, any time you visit a page with Drupal hard-coded content, it will be added into your language set database.

Once you have visited a page that you wish to change the content:

1. Go to the manage strings page (admin/locale/string/search) of the localization section.
2. Enter in the string you wish to search for.
3. Edit the result and enter your replacement text.

Menu: customize site navigation

The menu module allows for customization of the menus. Menus are useful for providing navigation in your site. The main menu for navigation is the navigation menu. Menus appear in blocks on your site.

- On the administer menu page administrators can "edit" to change the title, description, parent or weight of the menu item. Under the "operations" column, click on "enable/disable" to toggle the menu item on or off. Menu items which are disabled are not deleted; they are merely not available for navigating the site in the sidebar menu block. Note that the default menu items generated by the menu module cannot be deleted, only disabled.
- Using the "add menu" tab submit a title for a new custom menu. Once

submitted, the new menu will appear in a list toward the bottom of the administer menu page underneath the main navigation menu.

- Use the "add menu item" tab to create new links in either the navigation or a custom menu. Select the parent item to place the new link within an existing menu structure. For top level menu items, choose the name of the menu in which the link is to be added.
- To turn off the navigation menu block, administer the block page.

You can

- administer menus at administer >> menus
- turn menus blocks on and off in the administration >> block.
- add a menu at administer >> menus >> add menu.
- add a menu item at administer >> menus >> add menu item.

Node: the content

The node module manages the basic Drupal content type: the node or "post." All content on your site is stored in nodes or extensions of nodes such as blogs, stories, polls, forums, etc. (Note: comments are not stored as nodes but are always associated with a node). Having content as a node is an important aspect of making a content management system that can use new types of content and apply features to all content.

Node module features

- The list tab provides an interface to search and sort all content on your site.
- The configure settings tab has basic settings for content on your site.
- The configure content types tab lists all content types for your site and lets you configure their default workflow.
- The search tab lets you search all content on your site

You can

- search for content at search.
- administer nodes at administer >> content >> configure >> content types.

Page: post static pages

The page module allows users to create static pages, which are the most basic type of content. Pages are commonly collected in books via the book module. Users should create a page if the information on the page is static. An example would be an "about" page.

When a page is created, a user can set authoring information, configure publishing options, whether readers will be able to post comments. They can also select the content type of the page (e.g., full HTML, filtered HTML).

As an administrator, you can set the publishing default for a page (in its workflow): you can specify whether a page is by default published, sent to moderation, promoted to the front page, sticky at the top of lists, and whether revisions are enabled by default. You can set the permissions that different user roles have to view, create, and edit pages.

If the location module is enabled, then location specific information can be added. If the trackback module is enabled trackbacks can be configured.

You can

- read the node administration help at administer >> help >> node.
- read the page administration help at administer >> help >> page.
- create a page at create content >> page.
- administer page content type at administer >> content >> configure >> content types >> configure page.

Path: readable URL's

The path module allows you to specify aliases for Drupal URLs. Such aliases improve readability of URLs for your users and may help internet search engines to index your content more effectively. More than one alias may be created for a given page.

Some examples of URL aliases are:

- user/login => login
- image/tid/16 => store
- taxonomy/term/7+19+20+21 => store/products/whirlygigs
- node/3 => contact

The path module enables an extra field for aliases in all node input and editing forms (when users have the appropriate permissions). It also provides an interface to view and edit all URL aliases. The two permissions are related to URL aliasing are "administer a list of URL aliases" and "add url aliases".

This module also comes with user-defined mass URL aliasing capabilities, which is useful if you wish to uniformly use URLs different from the default. For example, you may want to have your URLs presented in a different language. Access to the Drupal source code on the web server is required to set up these kinds of aliases.

You can

- add a URL alias: administer >> url aliases >> add alias
- administer the list of URL aliases: administer >> url aliases

Mass URL aliasing

Drupal also comes with user defined mass URL aliasing capabilities. You might like to see completely different URLs used by Drupal, or even URLs translated to the visitors' native language, in which case this feature is handy. Only an administrator with access to the website source code can set up this kind of aliases. You can define a `conf_url_rewrite` function in `conf.php`, following this example:

```
function conf_url_rewrite($path, $mode = 'incoming') {
  if ($mode == 'incoming') { // URL coming from a client
    return preg_replace('!^display/(\d+)$!', 'node/\1', $path);
  }
  else { // URL going out to a client
    $aliased = preg_replace('!^node/(\d+)$!', 'display/\1', $path);
    if ($aliased != $path) { return $aliased; }
  }
}
```

This function will shorten every `node/$node_id` type of URL to `display/$node_id`. Individual URL aliases defined on the browser interface of Drupal take precedence, so if you have the 'contact' page alias from the example above, then the `display/3` alias will not be effective when outgoing links are created. Incoming URLs however always work with the mass URL aliased variant. Only the 'incoming' and 'outgoing' modes are supposed to be supported by your `conf_url_rewrite` function.

You cannot only use this feature to shorten the URLs, or to translate them to your own language, but also to add completely new subURLs to an already existing module's URL space, or to compose a bunch of existing stuff together to a common URL space. You can create a news section for example aliasing nodes and taxonomy overview pages falling under a 'news' vocabulary, thus having `news/15` and `news/sections/3` instead of `node/15` and `taxonomy/term/3`. You need extensive knowledge of Drupal's inner workings and regular expressions though to make such advanced aliases.

Ping: notify services of changes

The ping module is useful for notifying interested sites that your site has changed. It automatically sends notifications (called "pings") to the pingomatic service to tell it that your site has changed. In turn pingomatic will ping other services such as weblogs.com, Technorati, blo.gs, BlogRolling, Feedster.com, Moreover, etc.

The ping module requires cron or a similar periodic job scheduler to be enabled.

You can enable or disable the ping module at administer >> modules.

Poll: community voting

The poll module can be used to create simple polls for site users. A poll is a simple multiple choice questionnaire which displays the cumulative results of the answers to the poll. Having polls on the site is a good way to get instant feedback from community members.

Users can create a poll. The title of the poll should be the question, then enter the answers and the "base" vote counts. You can also choose the time period over which the vote will run.

The poll item in the navigation menu will take you to a page where you can see all the current polls, vote on them (if you haven't already) and view the results.

You can

- view the polls page.
- administer >> content >> configure >> poll.

Profile: extending user account information

The profile module allows you to define custom fields (such as country, real name, age, ...) in the user profile. This permits users of a site to share more information about themselves, and can help community-based sites to organize users around profile fields.

The following types of fields can be added to the user profile:

- single-line textfield
- multi-line textfield
- checkbox
- list selection
- freeform list
- URL
- date

You can

- view user profiles
- administer profile settings: administer >> settings >> profiles

Enabling user pictures (avatars)

Note that user pictures (or avatars) or pictures are part of the user.module, not the profile module.

1. Navigate to the administration area for user module configuration page under *administer Â» users Â» configure*

2. In the Pictures settings, for *Picture support*, select *Enabled* (*Enable picture support*.)
3. Note: you must write in your directory name. Make sure the directory is created and make sure you have that directory has write permissions (that is, chmod to 755).
4. Click 'save configuration'

HOWTO: Create new profile fields

1. Enable the profile module:
2.
 - In *administer >> modules* select in the 'Enable' column where you see: profile | Support for configurable user profiles.
3. Create new fields or edit existing ones:
4.
 - Navigate to the administration area: *administer >> settings >> profiles*
 - Select a form field type under 'Add new field'
 - Follow onscreen instructions for configuring this field, and 'save field'.

Search: an internal site search system

The search module adds the ability to search for content by keywords. Search is often the only practical way to find content on a large site.

The search engine works by maintaining an index of the words in your site's content. You can adjust the settings to tweak the indexing behaviour. Note that the search requires cron to be set up correctly. The index percentage sets the maximum amount of items that will be indexed in one cron run. Set this number lower if your cron is timing out or if PHP is running out of memory.

You can

- read about how your site uses cron in the [administer >> help >> system](#).
- run [your >> cron.php](#).
- [administer >> search](#).

Statistics: tracking referrers, page hits, etc.

The statistics module keeps track of numerous statistics of site usage. It counts how many times, and from where each of your posts is viewed. The statistics module can be used to learn many useful things about how users are interacting with each other and your site.

Statistics module features

- Logs shows statistics for how many times your site and specific content on your site has been accessed.
- Referrers tells you from where visitors came from (referr url)

- Top pages shows you what the most popular content on your site is
- Top users shows you the most active users for your site
- Recent hits displays information about the latest activity on your site
- Node count displays the number of times a node has been accessed in the node's link section next to # comments.
- Popular content block creates a block that can display the day's top viewed content, the all time top viewed content, and the last content viewed.

Configuring the statistics module

- enable access log allows you to turn the access log on and off. This log is used to store data about every page accessed, such as the remote host's IP address, where they came from (referrer), what node they've viewed, and their user name. Enabling the log adds one database call per page displayed by Drupal.
- discard access logs older than allows you to configure how long an access log entry is saved, after which time it is deleted from the database table. To use this you need to run *cron.php*
- enable node view counter allows you to turn on and off the node-counting functionality of this module. If it is turned on, an extra database query is added for each node displayed, which increments a counter.
- display node view counters allows you to globally disable the displaying of node view counters.

You can

- administer statistics administer >> settings >> statistics.
- access statistics logs administer >> logs >> recent hits.
- enable 'popular content' block in block administration administer >> blocks .

Old page

Introduction

The statistics module keeps track of numerous statistics for your site but be warned, statistical collection does cause a little overhead, thus everything comes **disabled** by default.

The module counts how many times, and from where -- using HTTP referrer -- each of your posts is viewed. Once we have that count the module can do the following with it:

- The count can be displayed in the node's link section next to "# comments".
- A configurable block can be added which can display a configurable number of the day's top stories, the all time top stories, and the last stories read.
- A configurable user page can be added, which can display the day's top stories, the all time top stories, and the last stories read. You can individually configure how many posts are displayed in each section.

Notes on using the statistics:

- If you enable the view counters for content, this adds 1 database query for each node that is viewed (2 queries if it's the first time the node has ever been viewed).
- If you enable the access log, this adds 1 database query for each page that Drupal displays. Logged information includes: HTTP referrer (if any), node being accessed (if any), user ID (if any), the IP address of the user, and the time the page was viewed.

As with any new module, the statistics module needs to be enabled before you can use it. Also refer to the permissions section, as this module supports four separate permissions.

Configuring the statistics module

There are some configuration options added to the main administer → settings → statistics section:

- *enable access log* -- allows you to turn the access log on and off. This log is used to store data about every page accessed, such as the remote host's IP address, where they came from (referrer), what node they've viewed, and their user name. Enabling the log adds one database call per page displayed by Drupal.
- *discard access logs older than* -- allows you to configure how long an access log entry is saved, after which time it is deleted from the database table. To use this you need to run "cron.php"
- *enable node view counter* -- allows you to turn on and off the node-counting functionality of this module. If it is turned on, an extra database query is added for each node displayed, which increments a counter.
- *display node view counters* -- allows you to globally disable the displaying of node view counters.

Popular content block

This module creates a block that can display the day's top viewed content, the all time top viewed content, and the last content viewed. Each of these links can be enabled or disabled individually, and the number of posts displayed for each can be configured with a drop down menu. If you disable all sections of this block, it will not appear.

Story: post static pages

The story module is used to create a content post type called *stories*. Stories are articles in their simplest form: they have a title, a teaser and a body. Stories are typically used to post news articles or as a group blog.

The story administration interface allows for complex configuration. It provides a submission form, workflow, default view permission, default edit permission, permissions for permission, and attachments. Trackbacks can also be enabled.

You can

- post a story at create content >> story.
- configure story at administer >> content >> configure types >> story configure.

System: cron and caching

The system module provides system-wide defaults such as running jobs at a particular time, and storing web pages to improve efficiency. The ability to run scheduled jobs makes administering the web site more usable, as administrators do not have to manually start jobs. The storing of web pages, or caching, allows the site to efficiently re-use web pages and improve web site performance. The settings module provides control over preferences, behaviours including visual and operational settings.

Some modules require regularly scheduled actions, such as cleaning up logfiles. Cron, which stands for chronograph, is a periodic command scheduler executing commands at intervals specified in seconds. It can be used to control the execution of daily, weekly and monthly jobs (or anything with a period measured in seconds). Automating tasks is one of the best ways to keep a system running smoothly, and if most of your administration does not require your direct involvement, cron is an ideal solution.

There is a caching mechanism which stores dynamically generated web pages in a database. By caching a web page, the system module does not have to create the page each time someone wants to view it, instead it takes only one SQL query to display it, reducing response time and the server's load. Only pages requested by *anonymous* users are cached. In order to reduce server load and save bandwidth, the system module stores and sends cached pages compressed.

You can

- activate your cron job on the cron page.
- administer Cache settings administer >> settings.

Configuring cron jobs

Some modules require regularly scheduled actions, such as cleaning up logfiles. Cron, which stands for chronograph, is a periodic command scheduler executing commands at intervals specified in seconds. It can be used to control the execution of daily, weekly and monthly jobs (or anything with a period measured in seconds). Automating tasks is one of the best ways to keep a system running

smoothly, and if most of your administration does not require your direct involvement, cron is an ideal solution.

The recommended way to set up your cron system is to set up a Unix/Linux crontab entry (see "man crontab") that frequently visits

<http://example.com/cron.php>. Note that cron does not guarantee the commands will be executed at the specified interval. However, Drupal will try its best to run the tasks as close to the specified intervals as possible. The more you visit cron.php, the more accurate cron will be.

If your hosting company does not allow you to set up crontab entries, you can always ask someone else to set up an entry for you. After all, virtually any Unix/Linux machine with access to the internet can set up a crontab entry to frequently visit <http://example.com/cron.php>. If you cannot setup cron yourself or find someone to do it for you, consider the poormanscron module.

For the Unix/Linux crontab itself, use a browser like lynx or wget but make sure the process terminates: either use `/usr/bin/lynx -source`

`http://example.com/cron.php` or `/usr/bin/wget -o /dev/null -O /dev/null <a`

`href="http://example.com/cron.php">http://example.com/cron.php`.

Take a look at the example scripts in the scripts-directory. Make sure to adjust them to fit your needs. A good crontab line to run the cron script once every hour would be:

```
00 * * * * /home/www/drupal/scripts/cron-lynx.sh
```

Note that it is essential to access cron.php using a browser on the web site's domain; do not run it using command line PHP and avoid using localhost or 127.0.0.1 or some of the environment variables will not be set correctly and features may not work as expected.

Taxonomy: categories and classification schemes

The taxonomy module is one of the most popular features because users often want to create categories to organize content by type. It can automatically classify new content, which is very useful for organizing content on-the-fly. A simple example would be organizing a list of music reviews by musical genre.

Taxonomy is also the study of classification. The taxonomy module allows you to define vocabularies (sets of categories) which are used to classify content. The module supports hierarchical classification and association between terms, allowing for truly flexible information retrieval and classification. The taxonomy module allows multiple lists of categories for classification (controlled vocabularies) and offers the possibility of creating thesauri (controlled vocabularies that indicate the relationship of terms) and taxonomies (controlled vocabularies where relationships are indicated hierarchically). To delete a term choose *edit term*. To delete a vocabulary, and all its terms, choose *edit vocabulary*.

A controlled vocabulary is a set of terms to use for describing content (known as descriptors in indexing lingo). Drupal allows you to describe each piece of content (blog, story, etc.) using one or many of these terms. For simple implementations, you might create a set of categories without subcategories, similar to Slashdot's sections. For more complex implementations, you might create a hierarchical list of categories.

You can

- administer >> taxonomy >> add vocabulary.
- administer >> taxonomy.

Vocabularies and terms

Each category group, or *vocabulary*, can contain multiple category entries, or *terms*, for tagging content.

For example, a web-based discussion community might have a vocabulary *Topics* with terms such as

- Technology
- Politics
- Education
- Religion
- Sports

An administrator might also choose to create multiple vocabularies for use with the same node type. Consider another vocabulary for use alongside of Topics, one which classifies nodes in another way:

Content with terms

- News
- Reviews
- Announcements
- Opinions

New vocabularies can also be created or added to at any time, with as few or as many terms as the administrator may need. And do not worry. Long before reaching Drupal's limits at handling very large classification schemes, users would find large vocabularies and terms unwieldy to use and maintain.

NOTE: When creating terms for a new vocabulary, administrators might want to provide users with a catchall term, such as *Miscellaneous*. Administrators can then review nodes tagged with Miscellaneous to see if a need exists for new terms. Once new terms are created, ambitious administrators can also update nodes with the new tag and remove the catchall category tag.

Creating a vocabulary

When setting up a vocabulary, Drupal will prompt for:

- **Vocabulary name** (Required) -- A name for this vocabulary; for example, Topics.
- **Description** (Optional) -- A description of the vocabulary (this item may be used by some modules and feeds).
- **Types** (Required) -- A vocabulary may be associated with or more node types. So, an administrator might declare that a particular vocabulary is to be associated with stories and blogs, but not book pages. If an expected node type is unavailable, check and make sure that the module for the specific node type has been activated.
- **Related terms** (Optional) -- Allows relationships between terms within this vocabulary. Think of these as "see also" references (this item is not used by many Drupal modules).
- **Hierarchy** (Optional) -- Allows a tree-like taxonomy (see Using Hierarchies below).
- **Multiple select** (Optional) -- Allows users to categorize nodes by more than one term. Useful for cross-indexing content. Nodes may then appear on multiple taxonomy pages.
- **Required** (Optional) -- Requires a user to select a term in this vocabulary in order to submit the node. Otherwise, when creating a node, users will be offered a *none* option as the default for each vocabulary.
- **Weight** (Optional) -- Allows the administrator to set the priority of this vocabulary when listed with other vocabularies. When vocabularies are left with the default weight of zero, Drupal displays multiple vocabularies in alphabetical order. Increasing a vocabularies weight with respect to other vocabularies will cause it to appear after them in lists. Conversely, lighter vocabularies will float nearer the top of lists. Useful for specifying which vocabulary a user sees first when creating a node.

Creating terms

Once you have finished defining the vocabulary, you may populate it with terms. When creating a term, note that the available options may depend on what was selected for related terms, hierarchy and multiple select when creating the vocabulary:

- **Term name** (Required) -- The name for this term. Example: Technology.
- **Description** (Optional) -- Description of the term (this item may be used by some modules and feeds).
- **Parent** (Required) -- Select the term under which this term is a subset -- the branch of the hierarchy that this term belongs under (only required when hierarchy is enabled for the vocabulary).
- **Synonyms** (Optional) -- Enter synonyms for this term, one synonym per line.

Synonyms can be used for variant spellings, acronyms, and other terms that have the same meaning as the added term, but which are not explicitly listed in this thesaurus, i.e. unauthorized terms (this item not used by many Drupal modules).

- **Weight** (Optional) -- The weight is used to sort the terms of this vocabulary (see explanation of weight above).

Advanced taxonomies: using hierarchies

For many users needing simple classification schemes, the examples above may be the only structure necessary for tagging site content. For more elaborate classification needs, consider the hierarchy option when creating vocabularies. Hierarchies allow the creation of sophisticated taxonomies with categories and subcategories in a tree structure, much like Yahoo categories or subject classifications used by libraries.

For example, the vocabulary *Food* could include the following categories and subcategories:

- Dairy
 - Milk
- Drink
 - Alcohol
 - Beer
 - Wine
 - Pop
 - Milk
- Meat
 - Beef
 - Chicken
 - Lamb
- Spices
 - Sugar

Note that the term Milk appears within both Dairy and Drink. This is an example of multiple parents for a term. Just select both parents when creating the term Milk.

Don't forget that the order of term siblings (e.g. Beef, Chicken, Lamb) can be controlled with the weight option.

Using vocabularies for navigation

When displaying nodes, both in teaser listings on the Drupal home pages and in full, single-node view, many Drupal themes display the categories applied to the node. If the user selects any category term, Drupal will then display a browsable listing for all nodes tagged with that term.

Examine the Taxonomy URL for one such category listing. The end of the URL should look something like this:

taxonomy/page/or/1

And another Taxonomy URL, for a different term, something like this

taxonomy/page/or/2

Note that Taxonomy URLs always contain one or more Term IDs at the end of the URL. These numbers, 1 and 2 above, tell Drupal which categories to display.

Now combine the Term ID's above in one URL using a comma as a delimiter

taxonomy/page/or/1,2

The resulting listing includes all nodes tagged with either term. Want to combine more categories? Just add more commas and numbers. Know that you can use the taxonomy section in Drupal site administration to find out any Term ID. Just place the cursor over any *edit term* and look to the status bar at the bottom of the browser. Then substitute the new Term ID's found there to create a different category listing.

Sometimes, listing all nodes for either term returns more than a user may need. A user might only be looking for nodes which exist in both categories only. To create a boolean "AND" listing, change the querystring parameter from "or" to "and":

taxonomy/page/and/1,2 .

In addition to displaying Drupal nodes by category on site, Drupal has category specific RSS feeds for other sites to access your site content. See how the URL format for the RSS feed is very similar to the Taxonomy URL:

taxonomy/feed/or/1,2

Built like a Taxonomy URL, it starts with taxonomy/feed, then has the query string parameter, and finally the term IDs.

Building individual Taxonomy URL's is not the most user friendly way to provide site users access to browseable listings. Nor do administrators necessarily want to build custom blocks for users with links to each category listing. To significantly extend the means of accessing nodes by category, download and install the optional taxonomy_html and taxonomy_dhtml modules from the Drupal downloads page. Each module provides a slightly different approach to creating vocabulary and term listings pages for users, as well as optional side blocks. Try both and decide which is best for users on your site. Either will certainly increase each site user's ability to browse content

More about Taxonomy

Taxonomy is more than just a module in Drupal. It is also the study of classification and a research area of information science in the digital age. Drupal administrators who want to push the limits of the Drupal taxonomy system might want to read about classification theory and application, as well as how it applies to Drupal taxonomy module development.

Creating a Block with links belonging to certain taxonomy terms

Question

war_boar wrote:

how to have a block of links to all terms which match taxonomy like Reviews: Anime Or Reviews: Movies without doing it manually

it should be titled, *Reviews* and underneath all blogs or stories which matched.

Answer

As a demo, see the block named *Physicians* at Internists.net

You will need to create a new Block of type=php. You will then want to paste in the code below, and customize the 'Physicians' subject and the \$tax array. \$tax the list of tids that you are interested in. The third element, named 'operator', can be *and* or *or*. So in your case, assuming the term ID for movies is '3' and the term ID for Anime is '6', you want:

```
<?php
// paste this code into a custom block of type=php
// customize the $tax array and the $subject as needed
$tax = array(1, 2);
$operator = "or";
$result = taxonomy_select_nodes($tax, $operator);
while ($obj = db_fetch_object($result)) {
  $node = node_load(array('nid' => $obj->nid));
  $items[] = l($node->title, "node/".$node->nid);
}
return theme('item_list', $items);
?>
```

Throttle: congestion control

The throttle module provides a congestion control throttling mechanism for automatically detecting a surge in incoming traffic. If the site gets linked to by a popular website, or otherwise comes under a "Denial of Service" (DoS) attack, your webserver might become overwhelmed. This mechanism is utilized by other modules to automatically optimize their performance by temporarily disabling CPU-intensive functionality. For example, in the site theme, you might choose to disable pictures when the site is too busy (reducing bandwidth), or in modules, you might choose to disable some complicated logic (reducing CPU utilization).

The congestion control throttle can be automatically enabled when the number of anonymous users currently visiting the site exceeds the specified threshold. The congestion control throttle can be automatically enabled when the number of authenticated users currently visiting the site exceeds the specified threshold.

You can

- enable throttle for modules at administer >> module.
- enable throttle for blocks at administer >> block.
- administer throttle at administer >> settings >> throttle.

Tracker: viewing new and updated content

The tracker module displays the most recently added or updated content to the website allowing users to see the most recent contributions. The tracker module provides user level tracking for those who like to follow the contributions of particular authors.

The "recent posts" page is available via a link in the navigation menu block and contains a reverse chronological list of new and recently-updated content. The table displays the content type, the title, the author's name, how many comments that item has received, and when it was last updated. Updates include any changes to the text, either by the original author or someone else, as well as any new comments added to an item. To use the tracker module to *watch* for a user's updated content, click on that user's profile, then the *track* tab.

You can

- view the most recent posts.
- view user profiles and select the track tab.
- not administer this module.

Upload: collaborate with files

The upload module allows users to upload files to the site. The ability to upload files to a site is important for members of a community who want to share work. It is also useful to administrators who want to keep uploaded files connected to a

node or page.

Users with the upload files permission can upload attachments. You can choose which post types can take attachments on the content types settings page. Each user role can be customized for the file size of uploads, and the dimension of image files.

You can

- administer user permissions at administer >> user >> configure >> permissions
- administer content at administer >> content types.
- administer upload at administers >> settings.
- .

Watchdog: monitor your site

The watchdog module monitors your system, capturing system events in a log to be reviewed by an authorized individual at a later time. This is useful for site administrators who want a quick overview of activities on their site. The logs also record the sequence of events, so it can be useful for debugging site errors.

The watchdog log is simply a list of recorded events containing usage data, performance data, errors, warnings and operational information. Administrators should check the watchdog report on a regular basis to ensure their site is working properly.

You can

- view watchdog logs at administer >> watchdog.
- view watchdog event logs at administer >> watchdog >> events.

User: access and management settings

The user module allows users to register, login, and logout. Users benefit from being able to sign on because it associates content they create with their account and allows various permissions to be set for their roles. The user module supports user roles which can setup fine grained permissions allowing each role to do only what the administrator wants them to. Each user is assigned to one or more roles. By default there are two roles *anonymous* - a user who has not logged in, and *authenticated* a user who has signed up and who has been authorized.

Users can use their own name or handle and can fine tune some personal configuration settings through their individual my account page. Registered users need to authenticate by supplying either a local username and password, or a remote username and password such as a Jabber ID, DelphiForums ID, or one from a Drupal powered website. A visitor accessing your website is assigned an unique ID, the so-called session ID, which is stored in a cookie. For security's sake,

the cookie does not contain personal information but acts as a key to retrieve the information stored on your server.

You can

- read user profile help at `administer >> help >> profile`.
- read about distributed authentication in the system module help at `administer >> help >> system`.
- administer user at `administer >> user`.

Managing access control with permissions and user roles

Roles, a way of assigning specific permissions to a group, allow you to fine tune the security, use and administration of Drupal. Users assigned to the role, or group, are granted those permissions assigned to the role. Common examples of roles used with which you may be familiar include: anonymous user, authenticated user, moderator, and administrator.

By default, Drupal automatically defines two roles as a part of site installation:

- **anonymous user** -- readers of the site who are either do not have an account or are not logged in.
- **authenticated user** -- the role assigned to new accounts on a Drupal site.

The anonymous user role should typically have the least access to the site of all roles. Authenticated users, because they took the time to register, might be given more permissions, such as the ability to create some types of content. If administrator approval is required for new users, or if they match certain criteria (such as having a company email address), you may be able to grant more permissions that way.

The first Drupal account created on a new installation, sometimes referred to as the "root user", always has full permissions for all Drupal activities, including administration and content creation, editing and removal.

More trusted users might be granted special privileges through an administrator-created role, and must be manually added to that role through the user administration interface. To create new roles:

1. Click `administer >> access control >> roles` tab.
2. Enter a label for the new role in the available text field at the bottom of the current list of roles.
3. Once you've added the role, select the *permissions* tab.
4. Your new role will be listed as a new column in the permission matrix. Grant permissions to the new role.
5. To add users to this role you will need to edit individual user accounts. Click `administer >> users` and the *edit* link for the user you wish to add to the role.

Then you can add this user to your new role under the *Roles* section of the user edit page.

Assigning permissions and users to roles

Access to almost all Drupal modules can be controlled by either enabling or disabling permissions for a given role. As a security precaution, the anonymous and authenticated users are configured with very minimal permissions during a site install. You'll have to consider which permissions to enable.

Go to the permissions administration page (*administer >> access control >> permissions* tab) to begin enabling or disabling permissions. Consider the following descriptions of permissions:

- Administer -- Administer permissions, such as *administer content* and *administer users*, are usually reserved for the most trusted site users. These administration privileges grant users extensive control of the specific module(s) described by the permission title. For example, when administer permissions are granted on modules associated with specific node types, the user will be able to edit and delete all content for that node type on the entire site. Reminder: you'll have to assign access administration pages rights to any role which also needs to configure site options in the administration menu.
- Access -- Permissions which grant access allow users read-only rights or general use of specific site modules, without any significant configuration privileges. Typically, these roles do not permit the creation of content. Most access permissions are safe to assign to any user role, although giving *access administration* should generally be reserved for the most trusted users.
- Create -- Allows users to create, but not necessarily edit later, the specified type of content. Generally applies to node types.
- Maintain -- These permissions generally enable a user to create content, as well as allowing the author of the submitted content to edit their own content. If you want to allow new site members to keep a weblog or work on the collaborative book, you'll need to enable maintain permissions for the authenticated user.

Adjusting permissions after adding modules

Whenever a module is enabled, even if it is merely turned off and on, permissions for that module are unassigned to all roles. As a security precaution, an administrator always needs to assign permissions to roles any time a module is enabled.

User authentication

Registered users need to authenticate by supplying either a local username and password, or a remote username and password such as a jabber, Delphi, or one from another Drupal website. See distributed authentication for more information on this innovative feature.

The local username and password, hashed with Message Digest 5 (MD5), are stored in your database. When you enter a password it is also hashed with MD5 and compared with what is in the database. If the hashes match, the username and password are correct.

Once a user authenticated session is started, and until that session is over, the user won't have to re-authenticate. To keep track of the individual sessions, Drupal relies on PHP's session support. A visitor accessing your website is assigned an unique ID, the so-called session ID, which is stored in a cookie. For security's sake, the cookie does not contain personal information but acts as a key to retrieve the information stored on your server's side. When a visitor accesses your site, Drupal will check whether a specific session ID has been sent with the request. If this is the case, the prior saved environment is recreated.

User preferences and profiles

Each Drupal user has a profile, and a set of preferences which may be edited by clicking on the *my account* link. Of course, a user must be logged into reach those pages. There, users will find a page for changing their preferred time zone, language, username, e-mail address, password, theme, signature, homepage, and distributed authentication names. Changes made here take effect immediately. Also, administrators may make profile and preferences changes *administer Â» users* on behalf of their users.

Module developers are provided several hooks for adding custom fields to the user view/edit pages. These hooks are described in the Developer section of the Drupal Handbook. For an example, see the `jabber_user()` function in `/modules/jabber.module`.

Using distributed authentication

Distributed authentication

One of the more tedious moments in visiting a new website is filling out the registration form. Here at *drupal.org*, you do not have to fill out a registration form if you are already a member of Drupal. This capability is called *distributed authentication*, and is unique to Drupal, the software which powers *drupal.org*.

Distributed authentication enables a new user to input a username and password into the login box, and immediately be recognized, even if that user never registered at *drupal.org*. This works because Drupal knows how to communicate with external registration databases. For example, lets say that new user 'Joe' is already a registered member of Delphi Forums. Drupal informs Joe on registration and login screens that he may login with his Delphi ID instead of registering with *drupal.org*. Joe likes that

idea, and logs in with a username of joe@remote.delphiforums.com and his usual Delphi password. Drupal then contacts the *remote.delphiforums.com* server behind the scenes (usually using XML-RPC, HTTP POST, or SOAP) and asks: "Is the password for user Joe correct?". If Delphi replies yes, then we create a new *drupal.org* account for Joe and log him into it. Joe may keep on logging into *drupal.org* in the same manner, and he will always be logged into the same account.

Drupal

Drupal is the name of the software which powers *drupal.org*. There are Drupal web sites all over the world, and many of them share their registration databases so that users may freely login to any Drupal site using a single **Drupal ID**.

So please feel free to login to your account here at *drupal.org* with a username from another Drupal site. The format of a Drupal ID is similar to an email address: **username@server**. An example of a valid Drupal ID is **mwlily@www.drupal.org**.

Contributed modules

A large number of community created modules allow you to enhance your Drupal system. Download Drupal Modules.

You must install and configure each module individually on your machine. Each module download contains specific install instructions. Further description of specific modules' administration and use is contained within.

Buddylist: list your social network

Buddy list enables users to add buddies in their social network to their user account. Users can maintain a list of their buddy's and keep track of what their buddies are posting to the site. They can also track their buddy's buddy's and thereby exploring a social network.

You can add buddies via user profiles. The administrator must enable viewing users profiles to be able to use the buddy list option. On the View tab of the user profile, there is a Buddy list section. One of the buddy list actions is Add Buddy. Select this action to add a user to your buddy list. If a user is already in your buddy list, the add action will be to remove the buddy. Administrators can also enable the buddylist block. This block allows you to see a list of your buddies. An administrator can also enable the Friends Of A Friend (FOAF) module to allow for sharing of buddy lists between different social networking applications.

You can

- add a buddy by looking at user profiles.
- allow users to view profiles in administer >> access control >> permissions.
- enable the buddy list block at administer >> block.
- administer the buddy list block at administer >> settings >> buddylist.

Event: set times for content

The event module allows for any type of content to be event enabled, meaning content can have a start and end time, and appear in calendars. The ability to event enable any content type combined with the ability to create new types of content make it possible to create unlimited types of calendars. The ability to broadly event enable content will allow for creative applications combining information and real world events.

The administrator can decide which content types should be events for their site. In content type configuration, administrators can select the calendar view options: never, all views, or only views for this type. For example, this makes it possible to have a general calendar which shows all meetups and house parties in the same calendar, and have a separate calendar for rallies which only contains the rallies content type. Calendars can be customized to view a specific content type or a category of content, using taxonomies.

Administrators can also set two types of options for events; general event options, and event overview options. General event options are for timezone configuration, time notation formats, and event block configuration. Event overview options allow calendar and table event default views. Administrators can also set general filter controls for content types and categories, via the event taxonomy controls.

You can

- enable content types to be event enabled at administer >> content >> content types >> configure type.
- administer general event options administer >> settings >> events.
- create content and set a start and end time, if the administrator has set that content type to be event enabled.

Flexinode: new content types

The flexinode module allows administrators to create simple new content types. Administrators find it very useful to create new types of content without having to program a new content module.

When creating a new flexinode, administrators are presented with a flexinode form to create their new content type. Once administrators have created their flexinode they can accept the format or choose to theme the content type to change its presentation. For users, creating content that is a flexinode is just like adding other content. The flexinode content type will show up alongside all normal content.

You can

- create a flexinode content type at administer >> content types >> add_type.
- create content >> add type.
- administer flexinode at administer >> settings >> flexinode.
- .

FOAF: friends of a friend

FOAF, or *Friend of a Friend*, refers to a set of *social networking* standards and tools for online sharing and searching of user profile information. Users will find this useful if they want to make a list of their friends and be able to export or import their list among social networking applications.

The FOAF module allows users to export FOAF documents based on their profile information. It also allows users to import profile information from external FOAF files, and even includes an option to let profiles auto sync when using distributed authentication. The FOAF module can export your buddy list if the buddy list module is enabled. You can download FOAF files in the view tab of a user profile if FOAF is enabled.

You can

- read the FOAF wikipedia explanation.
- read the FOAFNet specification.
- view user profiles and select a user to see their FOAF file.
- enable the buddy list module at administrator >> modules.

Fontsize: make text larger

The font size module allows users to change the font size of the site they are viewing. This is useful for creating 508 accessible websites.

The font size module adds a block which has two buttons, one for a larger text size and one for a smaller or normal text size. Javascript is used to load generic stylesheets that work with most themes and templates.

You can

- Enable the font size block at administer >> block >> fontsize.

Forms: forms for modules

The forms module is a behind the scenes helper module. It is intended as a generic form building module, allowing module developers to include forms to be created by administrators.

If the form module is enabled for the survey module, a user could create custom form elements for their surveys through a web interface.

You can not administer this module.

HTMLarea

Allows users to write custom html and formating of text using a graphical user interface.

It has the following tabs: htmlarea, toolbar, font names, font size, format block, plugins, custom js.

You can

- administer >> settings >> htmlarea.

Image: galleries of images

The image module is used to create and administer images for your site. Each image is stored as a post, with thumbnails of the original generated automatically. There are two default thumbnail sizes, thumbnail and preview. The thumbnail size is shown as the preview for image posts and when browsing image galleries. The preview is the default size when first displaying an image node.

Image administration allows the image directory and the image sizes to be set.

Image galleries are used to organize and display images in galleries. The list tab allows users to edit existing image gallery names, descriptions, parents and relative position, known as a weight. The add galleries tab allows you to create a new image gallery defining name, description, parent and weight.

You can

- view image handling messages in administer >> settings.
- view lists and add image galleries at administer >> image >> galleries.
- configure image sizes and file directories at administer >> settings >> images.

Interwiki: wiki syntax for linking

The interwiki module is the way to create links to the many wiki webs on the world wide web. Users avoid pasting in entire URLs, as they would for regular web pages, and instead use a shorthand similar to links within the same wiki. The types of interwiki links allowed in a wiki are defined by an InterMap.

The interwiki module allows you to use interwiki links that point to wikis such as Wikipedia.org, SourceWatch.org and dKosopedia.com. It can also be used to link easily to Google and eBay searches and other online reference sources such as the Merriam-Webster dictionary. It uses a table interwiki which is the same table used by MediaWiki. MediaWiki users should be able to use an interwiki table interchangeably. The URL filter module must be enabled to use interwiki.

You can

- read the wikipedia interwiki definition.
- enable the urlfilter module in administrater >> module.
- enable the interwiki filter in your administration >> filter, input formats.
- administer interwiki administer >> interwiki .

Listhandler: connect mailing lists to forums

The listhandler module allows you to connect mailing lists to forums and vice versa. It works in conjunction with the mailhandler module. Mailhandler receives an email and then asks listhandler if the received email is part of a list. If the email is from a mailing list associated with a forum on your site, then listhandler adds the received email to the forum.

Listhandler administration allows you to set the email address of the list administrator. The email address is used as the From: field to check the address of messages sent by anonymous users. You can also enable and disable the listhandler for a list.

You can:

- administer listhandler administer >> listhandler.
- administer mailhandler administer >> mailhandler.
- administer forum administer >> forum.

Location: associate geographic location

The location module allows you to associate a geographic location with content and users. Users can do proximity searches by postal code. This is useful for organizing communities that have a geographic presence.

To administer locative information for content, use the content type administration page. To support most location enabled features, you will need to install the country specific include file. To support postal code proximity searches for a particular country, you will need a database dump of postal code data for that country. As of June 2005 only U.S. postal codes are supported.

You can

- administer locative information at administer >> content types to configure a type and see the locative information.
- administer location at administer >> settings >> location.
- use a database dump for a U.S. postal codes table that can be found at zipcode database.

Mailalias: alias e-mails to user

The mail alias module associates additional e-mail aliases to user accounts. The mailhandler module uses these email aliases to authenticate incoming e-mail with your account. This is useful because many users have multiple email accounts and administrators want to aggregate their content submitted by e-mail to the single user account.

The module is enabled on the users account settings page as a text field. e-mail aliases: where users can enter multiple e-mail addresses separated by a comma.

You can

- view the user page at user.
- administer users at administer >> users.
- administer mailhandler at administer >> mailhandler.

Mailhandler: content via mail

The mailhandler module allows registered users to create or edit nodes and comments via e-mail. Users may post taxonomy terms, teasers, and other post attributes using the mail commands capability. This module is useful because e-mail is the preferred method of communication by community members.

The mailhandler module requires the use of a custom mailbox. Administrators can add mailboxes that should be customized to meet the needs of a mailing list. This mailbox will then be checked on every cron job. Administrators may also initiate a manual retrieval of messages.

This is particularly useful when you want multiple sets of default commands. For example , if you want to authenticate based on a non-standard mail header like Sender: which is useful for accepting submissions from a listserv. Authentication is usually based on the From: e-mail address. Administrators can edit the individual mailboxes when they administer mailhandler.

You can

- run the cron job at cron.php.
- add a mailbox at administer >> mailhandler >> add a mailbox.
- administer mailhandler at administer >> mailhandler.

Massmailer: manage mailing lists

The massmailer module allows you to create newsletter style mailing lists that visitors to your website can subscribe to. It provides an interface to then email your subscribers in bulk, manage your subscription lists, and create and delete lists. The mass mailer module is built in such a way that it is possible to swap out the mass mailing back-end or *engines*. Currently the only usable engine is phplist. However, this module has aspirations for great mailing engines.

Mass mailer requires a mailing engine such as phplist to be installed and enabled. Configuring phplist can be difficult and requires settings to be set up correctly on your webserver.

You can

- read about phplist.
- administer massmailer at administer >> settings >> massmailer.
- create and configure mailing lists at administer >> mass mailer.

Node import: get CSV content

The node import module enables importing of nodes of any type into your site using comma separated values format (CSV). One possible use is with contact manager to import lists of contacts. Users want to be able to import content from other systems into their site.

Node import accepts a CSV file as input. CSV files can be generated using spreadsheet programs. Your CSV file must contain field names in its first row. These field names can be anything; except when using raw data import type. Modules, such as contact_manager, will add additional import types.

You can

- read about the function that reads CSV files called fgetcsv.
- import nodes at administer >> content >> import.
- administer node permissions at administer >> access >> permissions >> node import.

Raw data import type

The raw data import type allows control over what the content of each node object but requires specially formatted CSV files.

Your CSV file must contain field names in its first row. The following fields are required:

- 'title' the title of the post.
- 'uid' (user ID) or 'name' (user name).
- 'type' the type of the node.

UID or name are ignored if current user does not have administer nodes permission. If the user does have this permission, name is used if available, otherwise UID, otherwise anonymous user is assumed. Your CSV file should use quotes around strings that contain a comma. Parsing is handled by the php fgetcsv function. Field names should match those usually entered into your database.

Node privacy by role: node view and edit permissions

The node privacy by role module allows users, when creating or editing a post, to select which roles of users on a site will have *view* permissions for the node and which users on a site will have *edit* permissions. Community leaders frequently want to give permissions to roles to create and manage content for a site.

The node privacy by role module must be explicitly enabled in its administration interface. The node privacy by-role permissions are set by users for their nodes. If the node privacy by role module is disabled, the default permissions scheme will be in effect again, in which all users have view permissions for all nodes. However, if the module is re-enabled, the node-by-node permissions that were in place during the previous period in which the module was enabled will take effect again. Roles given edit permissions are automatically given view permissions even if the user tries to give *edit* permissions to a particular role, but not view permissions.

You must explicitly enable or disable the module at administer >> settings >> node_privacy_byrole.

You can

- set default permissions for each content type in the default workflow settings at administer >> content >> configure >> content types for each content type.
- decide who can ignore the default permissions for each content type in the default workflow settings at administer >> content >> configure >> content types for each content type.

Notify: email notification of new site content

The notification module allows users to subscribe to periodic emails which include all new or revised content and/or comments much like the daily news letters sent by some websites. Even if this feature is not configured for normal site users, it can be a useful feature for an administrator of a site to monitor content submissions and comment posts.

The administrator sets the frequency of the emails in the notify administration interface. They can also set how many email failures should occur before notify stops sending notifications. Note that cron must be enabled for notifications to be sent out.

You can

- set up your site to run tasks automatically at required intervals. For more information, see cron.
- administer notify administer >> settings >> notify.

Privatemsg: an internal messaging system

The private messaging module allows users to send messages to each other without having to share email addresses. An inbox link will appear in the navigation menu. The "write to author" links are included in posts, allowing users to write a private message instead of commenting openly. Allowing users to communicate directly is an important part of building the strength of the community.

Users can also select whether to receive email notices of new messages by editing their user profile. The contacts list contains only users that you have previously messaged. To contact users not in your list, you need to know their local user name. Administrators can set messaging options such as frequency of emails, message status display, and number of messages to display per page. They can also configure 'Write to Author' options.

You can

- administer privatemsg at administer >> settings >> private message.

Queue: moderation, collaborative rating

The queue module allows community leaders to moderate content to meet the challenges of information overload. This assists users in identifying the most interesting, worthwhile, valuable or entertaining items.

Administrators can set the number of responses required before a post is promoted to the front page. They can also set whether comments are published. Administrators can also set conditions for posts to be unpublished.

You can

- administer queue at administer >> settings >> queue.

Moderation queue

Anyone who visits and has some news or some thoughts they'd like to share, can submit new content for consideration. After someone has submitted something, their node is added to a queue. All registered users can access this list of pending nodes, that is, nodes that have been submitted, but do not yet appear on the public front page. Those registered users can vote whether they think the node should be posted or not. When enough people vote to post a node, the node is pushed over the threshold and up it goes on the public page. On the other hand, when too many people voted to drop a node, the node will get trashed.

Moderation depends upon activation of the queue module. Users need the "access submission queue" before they are able to access the submission queue.

Comment rating

Anyone with a user account will be able to moderate comments. This lets people assign a score to a comment on how good they think the comment is or how visible they think it should be.

When more than one person rates a comment, the overall rating is just a simple average of all ratings. Comments with high ratings are more visible than comments with a lower rating. That way, comments that gain the approval of participants will gradually move up through statistical effects and pointless comments will sink into oblivion.

Hence, the purpose of comment moderation is two-fold:

- To bring the really good comments to everyone's attention.
- To hide or get rid of spam, flamebait and trolls.

In the latter, comment moderation provides a technical solution to a social problem.

RSVP: invite people

The RSVP module lets users invite people by email to events and track a list of people who will be attending. The RSVP module requires the event module because it is necessary to have an event to invite people to first.

The RSVP page shows a *your invites* and a *your RSVPs* tab. There are confirmation screens for creating and editing RSVPs. Email addresses which are input for RSVP have input validation. RSVP also creates an invitation url by hash value access so that users can click a URL and be taken directly to their invitation. For each RSVP there are view, edit, manage, and send tabs. Users can manage attendees through the manage attendees tab. Users can also send attendees a message through the send message tab.

You can

- view your RSVPs at RSVP.
- not administer the RSVP module.

Survey: community questions

The survey module allows users to create surveys to be completed by site visitors. Survey submissions are stored in the database, can be downloaded to Excel, and can generate an e-mail notification of each survey submission for the survey administrator. Surveys are valuable for gathering information about a community effectively.

Creating a survey is a two-step process. First create a new survey. A new survey requires title, thank-you page, survey notification e-mail, and intro text; attachments are optional. Once the survey is created, there are six tabs for

managing the survey: view, edit, outline, track, form, and responses.

From the edit tab, select *add survey question* to add questions. The outline feature allows you to include survey responses in the book hierarchy. Track allows you to see time, referrer, user, and operations in a log so that you can monitor how users are coming to your survey. Finally, when users have responded to your survey, responses can be seen on the survey page. This module requires the forms module to create survey forms.

You can

- create a survey at create content >> survey.
- adminsiter surveys at administer >> content types >> survey.

Taxonomy menu: navigation for terms

Taxonomy terms allow classification of content into categories and subcategories. The taxonomy menu module adds links to the navigation menu for taxonomy terms. This is useful when the community is focused on creating content that is organized in a taxonomy.

The taxonomy menu administration interface allows taxonomy terms to be enabled to show in the navigation menu. You can also select whether a term's descendants subterms are displayed.

You can

- view a list of taxonomies in administer >> taxonomy.
- create a new vocabulary at administer >> taxonomy >> add vocabulary.
- administer taxonomy_menu settings by going to administer >> settings >> taxonomy menu.

Textile: simple text syntax

The textile module allows users to enter content using Textile, a simple, plain text syntax that is filtered into valid XHTML. Textile enables users to learn to format content quickly without having to worry about more complex syntax of html or xhtml. The filter tips page provides syntax descriptions and examples. Users can use syntax to create:

- headings
- paragraphs
- block quotes
- ordered lists
- unordered lists
- footnotes
- tables

Textile editing is an Input Format option on content creation and editing forms. It allows forblockquote, br, and html tags. It also automatically converts e-mail and web addresses to active links.

You can

- read about textile.
- read about filters at filter >> tips.
- not administer this module.

Theme editor: store, configure, create

The theme editor module enables the administrator to edit the basic themes for the web site, delete a theme from the theme storage directory, rename a theme, and re-configure the storage directory.

Theme editor has a list, new, and configure tab. The list tab allows for editing, deletion, and renaming. The new tab allows for a theme to be copied to the theme storage directory for editing. The configure tab creates a sub-directory where theme files will be stored.

You can

- administer theme editor at administer >> theme editor.
- add a new theme to the theme storage directory from administer >> theme editor >> new theme editor.
- edit the theme storage directory from administer >> theme editor >> configure.

TinyMCE: a WYSIWYG editor

The TinyMCE module adds what-you-see-is-what-you-get (WYSIWYG) html editing to text areas. This enables users to create rich content easily.

TinyMCE profiles can be based on user roles. It can define which pages receive this TinyMCE capability, what buttons or themes are enabled for the editor, how the editor is displayed, and a few other editor functions. The default profile setting uses the "simple" TinyMCE theme which just shows the most minimal button set (bold, italic, underline, etc), but many other settings are available by switching to "advanced". And by default, textareas on affected pages will be automatically swapped out with the rich text editor. Users may disable the editor for any textarea without reloading the page. This setting can be reversed so that pages load with conventional textarea form fields. A link below each textarea allows TinyMCE to be turned on or off *on the fly*.

If a user is a member of roles defined in multiple profiles, they will receive the profile with the lowest role id they belong to.

You can

- administer roles at administer >> access control >> roles.
- change user TinyMCE account settings, at administer >> user >> edit.
- administer TinyMCE at administer >> settings >> tinymce.

Trackback: post remotely

The trackback module allows users to give a blog post a contextual link to another. A context is made because the trackbacking poster is, in theory, writing about something mentioned on another blogger's trackbacked post. Using the trackback URL accompanying each post, another website can send a ping to your website. Once received, the trackback will show up on the trackback page accompanying the post. It also includes auto-discovery, spam moderation queues, and the ability to manually ping another site.

If trackback autodiscovery is enabled on your website, someone need only visit your post via a link from another website post to have trackback *discover* the linking site and create the trackback. Trackback auto-discovery also works internally within a website, automatically creating connections between pages which link to each other. To manually send a ping to another site, edit your post and use the trackback URL field at the bottom of the edit page to submit the trackback URL for the post on the other site. Once you enter submit, your website will ping the other site for you. With trackback autodiscovery enabled, your site will attempt to do this automatically without your intervention.

To enable the moderation queue, go to the administer trackbacks page and select the configure tab. To view, approve, or delete trackbacks awaiting moderation, go to the administer trackbacks page and select the approval queue. To administer the trackback settings for a particular content type go to that content types administration page.

You can

- administer trackbacks at administer >> trackbacks.
- configure trackbacks at administer >> trackbacks >> configure trackbacks.
- administer trackbacks for content types at administer >> content >> configure >> content types.
- review your approval queue at administer >> trackbacks >> list >> approval queue.

URL filter: turn URLs and e-mail addresses into live links

The URLfilter module automatically converts text web addresses (URLs, e-mail addresses, ftp links, etc.) into hyperlinks. This is useful for users if content authors do not explicitly create a hyperlink for the URL.

Use Input Formats to enable the URL filter.

1. Select an existing Input Format or add a new one
2. Configure the Input Format
3. Enable URL filter and Save configuration
4. Rearrange the weight of the URL filter depending on what filters exist in the format

You can enable the urlfilter in an input format from administer >> Input Filter.

Volunteer: organize people to help

Volunteer module helps you organize people for events. There is a managed email correspondence with the volunteer, as well as the ability to rate their volunteer performance at the end. It allows admins to further designate users as *volunteer coordinators* who can be designated as the contact person for volunteers for an event.

This module currently relies on the CivicSpace contact manager module to handle contact information for volunteers. The volunteer module is frequently used with the RSVP module to invite community members.

You can

- administer >> settings >> volunteer.

Upgrading from previous versions

This chapter contains articles that discuss the upgrading process of your drupal installation.

please note that comments are not meant to address problems you fund during installation. For problems with installation, please address your questions at proper places

Upgrading from Drupal 2.00 to 3.00

Upgrading your Drupal database can be tedious and sometimes painful. I would like to make available to the Drupal community an update script to make this process easier. Please keep in mind it is not perfect and only one possible solution. I have also added some thoughts on the upgrade process in

general. This may also be useful if you are considering switching from another CMS to Drupal v3.

The script is available to download here:

<http://www.nodalpoint.org/downloads/update.tgz>

Notes on the script:

The update script needs to be run under Drupal rc2 since it includes various Drupal functions. In its current state it is only useful for upgrading from post node v2 Drupal. However the functions are simple enough that only minimal hacking should be required to upgrade from different table structures. Some brief instructions are included in the download.

Other considerations:

Users, roles and permissions

Users will need to be given a role when they are inserted into the database. Check the insert statement in the update_users() function. Read about roles and permissions [here](#).

Sections vs Meta-tags

In upgrading from sections to meta-tags, I chose to dump sections as meta-tags all belonging to the one collection. The script creates the collection. This is of course not the best way to utilize meta-tags, just a simple way of dumping sections.

If others have useful information regarding how to move from drupal2/nuke/phpslash/slash etc. to Drupal please submit a revised copy of this page or another book page. Lastly: remember the usual caveats about backing up your data!

Upgrading from Drupal 3.00 to 4.00 and later versions

Drupal 4.0 has an automatic upgrade script which upgrades your database from version 3.0 to 4.0. Point your browser to <http://yoursitename.com/update.php> and follow the instructions.

Note: same update script also allows to update your 4.0 database to the latest development version.

Migrating from other weblog software

Migrating from ezPublish

I've migrated an ezPublish site to drupal. Here are the steps:

1. I performed a basic installation of drupal, and created the first user.
2. I used phpMyAdmin to extract the entire ezPublish database, then installed it at the target site.
3. I used sql statements to extract articles, links, and users from ezPublish and insert them into drupal database.
4. I modified ezPublish's "printer-friendly" article template to insert html comments showing the start and end of teaser and body.
5. ezPublish maintains articles in ezxml. I used a perl script to fetch each ezPublish article with LWP::UserAgent, extract the html-formatted teaser and body, and update the content of the drupal database.
6. I used a perl script to extract user's first and last names from ezPublish and package them into the users.data field for use by drupal's profile module.

Here are the sql and perl scripts I used. Please note the following limitations:

1. Doesn't know how to access ezPublish pages that require login; the content of these pages will be left in ezxml.
 2. Doesn't do any content except articles, article categories, links, link categories, and users.
- Doesn't fix internal links (links to other pages on same site), but does identify nodes containing them.

Move ezb database content to drupal database

[note from editor ax: you have to escape the special characters < (<), > (>), and & (&)]

```
mysql -ppassword drupal < migrate.sql
```

The following is the content of migrate.sql:

```
select @uid := if(max(uid),max(uid),0) from users;
select @tid := if(max(tid),max(tid),0) from term_data;
select @nid := if(max(nid),max(nid),0) from node;
select @role_authenticated_user := rid from role where name = 'authenticated user';
select @ezp_url := "http://myezpsite.com";
#
# Insert all ezb publish articles as drupal "story" nodes
#
INSERT INTO node
(nid, type,title,uid,status,comment, promote, users, attributes, revisions, created,teaser,body)
select
id+@nid,          # nid
'story',           # type
name,              # title
1,                 # uid
1,                 # status
2,                 # comment
0,                 # promote
'',                #
'',                #
'',                #
```

```

created,
contents,
contents
from ezp.eZArticle_Article
where IsPublished;
#
# Insert all ezpublish weblinks as nodes
#
select @weblink_nid:= max(nid)+1 from node;
INSERT INTO node
(nid, type, title, uid, status, comment, promote, users, attributes, revisions, created, teaser, body)
select
id+@weblink_nid,
'weblink',
name,
1,
1,
2,
0,
 '',
 '',
 '',
 '',
created,
description,
description
from ezp.eZLink_Link;
INSERT INTO weblink
(
nid,
weblink,
click,
monitor,
#size,
change_stamp,
checked,
#feed,
refresh,
threshold,
spider_site
#spider_url
)
select
id+@weblink_nid,
if (url regexp '://', url, concat('http://', url)),
0,
0,
0,
0,
21600,
40,
0
from ezp.eZLink_Link;
#
# Discover vocabularies for ezarticle and one for ezlink
# (These established manually by drupal configure/taxonomy UI)
#
select @topic := vid from vocabulary where name = 'Topic';
select @link := vid from vocabulary where name = 'Link';
#
# Insert ezarticle_category names as terms under topics vocabulary
#
INSERT INTO term_data
(
tid,
vid,
name,
description,
weight
)
select
id+@tid,
@topic,
name,
description,
0
from ezp.eZArticle_Category;
#
# The article categories (terms) are non-hierarchical

```

```
#  
insert into term_hierarchy  
select id+@tid,0 from ezp.eZArticle_Category;  
#  
# Insert categories assigned to ezarticles  
#  
INSERT INTO term_node  
(  
nid,  
tid  
)  
select  
articleid+@nid,  
categoryid+@tid  
from ezp.eZArticle_ArticleCategoryLink  
;  
#  
# Insert eZLink_Category names as terms under vocabulary links  
#  
select @weblink_tid := max(tid)+1 from term_data;  
INSERT INTO term_data  
(  
tid,  
vid,  
name,  
description,  
weight  
)  
select  
id+@weblink_tid,  
@link,  
name,  
description,  
0  
from ezp.eZLink_Category;  
#  
# The link categories (terms) are non-hierarchical  
#  
insert into term_hierarchy  
select id+@weblink_tid,0 from ezp.eZLink_Category;  
#  
# Insert categories assigned to ezlinks  
#  
INSERT INTO term_node  
(  
nid,  
tid  
)  
select  
linkid+@weblink_nid,  
categoryid+@weblink_tid  
from ezp.eZLink_LinkCategoryLink  
;  
#  
# Insert users  
#  
INSERT INTO users  
(  
uid ,  
name ,  
pass ,  
mail ,  
# mode ,  
# sort ,  
# threshold ,  
# theme ,  
signature ,  
timestamp ,  
status ,  
timezone ,  
# language ,  
init ,  
# data ,  
rid  
)  
select  
id+@uid,  
login,
```

```

password, # encryption differs, so users will have to reset their passwords
email,
signature,
1074479825,
1, # active status
0, # timezone
email, # init
@role_authenticated_user
from ezp.eZUser_User;
#
# drupal declares these table primary keys as auto_increment, but
# in fact actually assigns them explicitly. Update drupal's idea
# of what id to assign next for each table.
#
delete from sequences where name='users_uid';
insert into sequences (name, id)
select 'users_uid', max(uid) from users;
delete from sequences where name='term_data_tid';
insert into sequences (name, id)
select 'term_data_tid', max(tid) from term_data;
delete from sequences where name='node_nid';
insert into sequences (name, id)
select 'node_nid', max(nid) from node;
#
# Identify articles with internallinks (to be edited manually in drupal)
#
select nid from node where body regexp @ezp_url;

```

Parse ezxml (in perl, with LWP::UserAgent)

```

#!/usr/bin/perl -w
use strict;
use LWP::UserAgent;
use HTTP::Request::Common qw(POST);

use DBI;
use Carp;
sub parse;
my $server = 'localhost';
my $database = 'drupal';
my $username = 'me';
my $password = 'foobar';
my $verbose;
my $dbh = DBI->connect("dbi:mysql:$database:$server", $username, $password )
    or croak "Can't connect to database";
$dbh->{RaiseError} = 1;
my $select = $dbh->prepare( q/select nid from node where type='story' / );
my $update = $dbh->prepare( q/update node set teaser=?, body=? where nid=? / );
$select->execute;
while (my $id = $select->fetchrow) {
    my $ezpid = $id;

    # The following is the "printer-friendly" url for ezp article
    my $url = "http://mathiasconsulting.com/article/articleprint/$ezpid/-1/0/";
    my $uri = URI->new( $url );
    my $ua = LWP::UserAgent->new();
    my $req = POST $uri, [ ];
    # Send the request, receive the response
    my $response = $ua->request($req)->as_string;
    #
    # print "*****\n", uc($url), "\n";
    # print "$response\n\n";
    ($my $teaser, my $body) = parse( $response );
    if ($teaser and $body) {
        $update->execute( $teaser, "$teaser\n<!--break-->\n$body", $id );
    }
    else {
        print "Can't parse $url\n";
    }
}

```

```

}
#
# Look for lines placed there by articleprint.tpl
#
sub parse {
    my $s = shift;
    my $teaser;
    my $body;
    if ($s =~ /<!-- teaser starts --&gt;\n(.*)<!-- teaser ends --&gt;\n/ms) {
        $teaser = $1;
    }
    if ($s =~ /<!-- body starts --&gt;\n(.*)<!-- body ends --&gt;\n/ms) {
        $body = $1;
    }
    return $teaser, $body;
}
</pre>

```

Get ezpublish user real names for drupal profile.module

[note from ax to cheryl: this code triggers "suspicious input" because it of "data=". had to escape this with "data=". i also wrapped the lines ("my \$template=") at 80 chars to make this look better here - hope i didn't introduce any bugs]

```

#!/usr/bin/perl -w
use strict;
use DBI;
use Carp;
my $server = 'localhost';
my $database = 'drupal';
my $username = 'me';
my $password = 'password';
my $verbose;
my $dbh = DBI->connect("dbi:mysql:$database:$server", $username, $password )
    or croak "Can't connect to database";
$dbh->{RaiseError} = 1;
# difference between ezb user id and drupal uid (see @uid in migrate.sql)
my $iddifference = 1;
my $template='a:13:{s:16:"profile_realname";s:%d:"%s";s:15:"profile_address";\
s:0:"";s:12:"profile_city";s:0:"";s:13:"profile_state";s:0:"";s:11:"profile_zip";\
s:0:"";s:15:"profile_country";s:0:"";s:11:"profile_job";s:0:"";s:16:"profile_homepage";\
s:0:"";s:17:"profile_biography";s:0:"";s:11:"weblink_new";s:1:"0";s:5:"pass1";\
s:0:"";s:5:"pass2";s:0:"";s:5:"block";a:0:{}';
my $select = $dbh->prepare( q/select ID, FirstName, LastName from ezb.eZUser_User/ );
my $update = $dbh->prepare( q/update users set data=? where uid=?/ );
$select->execute();
while ((my $id, my $first, my $last) = $select->fetchrow) {
    my $name = ($first || '') . ($first && $last ? ' ' : '') . ($last || '');
    my $profile = sprintf( $template, length( $name ), $name );
    $update->execute( $profile, $id+$iddifference );
}

```

Migrating from Geeklog

The partial migration of stories from Geeklog into story-nodes Drupal is a mapping of the *_stories table into the nodes table; a quick way to do the transform is to dump the stories out into a format suitable for load data infile:

```

select 'story' as type,
       title,
       unix_timestamp(date) as created,
       '' as users,
       introtext as teaser,
       bodytext as body,
       unix_timestamp(date) as changed,
       '' as revisions
  from tc_stories into outfile '/tmp/stories.dump';

```

this creates the load file; after you've loaded the database script from the Drupal distribution, this data can be inserted into the database with

```

load data infile '/tmp/stories.dump'
  into table node
  (type,title,created,users,teaser,body,changed,revisions);

```

This is not a perfect transformation, but it's a start. Geeklog subjects are lost: To preserve categories, you would need to pre-load the topics in Drupal, then create a script that would insert items from *_stories as mapped in the above example, but then to fetch the nid node id number from the newly inserted record and do a search on the term_data to get the tid number, then insert the pair into the term_node table.

Since the terms of our new site were only superficially similar to the categories we'd used in Geeklog, we chose instead to fix up the categories later by doing keyword searches on stories to get a list of nid and then pairing those to the new topics by hand using SQL via mysql

Bug: After inserting stories using the above method, the stories will be in the archives, but will not appear on the main page (use update node set promot = 1 to fix this for all or selected items). A more serious bug is that the nodes do not appear in search results -- I don't know that much about the inner workings of Drupal, but I expect someone will post a comment explaining how to fix this.

Migrating from LiveJournal

In some respects, there is no need to migrate from Livejournal, as such. It's great.. one thing I can't offer here is the 'friends' feature, and all the othe great stuff the LJ offers.

This posting is for people who wish to either include their LJ data in a Drupal site or to leave LJ behind and import their data wholesale into a Drupal Blog.

When I started playing with Drupal I tried to import my LiveJournal in several ways... You may be happy with one of these. These are listed in order of best integration with Drupal.

If anyone see a mechanism for making their export system any better 1) email me.
2) email livejournal..!!

Import your LJ through an IFRAME held in a book page or similar

Not ideal to be honest, although I did use this approach for a week or two. You will rely upon the LJ commenting feature, and you will have to create a suitable LJ style to make it work.

Interestingly, you may not be aware that you can, in fact, have as many styles as you wish. 'They' will not tell you this anywhere... but you can.

You could have one style for people who view your journal directly at livejournal.com, and another for your importing IFRAME.

You simply reference them (in an IFRAME) like this:

```
<!--      Setup journal      -->
<iframe src="http://www.livejournal.com/customview.cgi?
user=rowanboy&styleid=186838" width="100%" height="300" frameborder="0" scrolling="auto"
allowtransparency="false">
<!-- Alternate content for non-supporting browsers -->
      Your browser won't work here. You'll need IE5.5+
</iframe>
```

When you define your custom style at LJ, you'll be told the style id to use. Note that this will only work for paid users of LJ.

Using provided Import Module

You *could* use the 'import' module to connect to the RSS feed provided by LiveJournal for every PAID user. It's pretty good that LJ even bother with this, so I guess we have to be grateful..

A standard RSS feed is provided at a url similar to:
<http://www.livejournal.com/users/rowanboy/rss.xml>

This will work if imported into the newsfeeds section of your Drupal site....but the actual content will still live at LiveJournal. I guess the advantage here is that you can still use LJ and yet (fairly) dynamically pull content into Drupal.

Use the Livejournal Module to import the raw data into Drupal

Current Download: from my site

I've written a module to import an entire LiveJournal into a Drupal Blog.

This kind of assumes that you're willing to bin LiveJournal at present. I'm working on a more dynamic approach, but once you see how this works, you'll probably understand my motives.

The alternative to the RSS feed - which I'll reiterate has *no content* - is to export your LJ month by month to an XML file. You'll find what you need to do this here. Save the resulting XML somewhere. Repeat for each month that you've been using LJ. Sigh.

Note: The module I've written will expect you to post this file on the 'net while you import it.

In essence, the module takes this file and stuffs it into a user blog. You can then decide whether to publish your entries or not.

I prefered this approach as I now own my data. If LJ went bust, it's my data...

Migrating from Movable Type

I [ax: jibbajabbaboy] 've chronicled my experience migrating a MovableType site to Drupal.. The process was fairly simple, but required a bit of work setting up index templates to export MT data as a MySQL dump with INSERT statements. If you have MT installed with a MySQL database I presume there might be an easier way to do this, but this I worked this way because it was much quicker for me. The final migrated Drupal site is now live at <http://urlgreyhot.com>.

[ax: cowboyd used another strategy. moved here from a comment.] I used a different strategy, but I didn't have the need to do anything fancy, such as bring over MT categories or comments, so I wrote a really quick perl script to parse the MT rss feed, and use it to directly generate inserts into the node table. Worked quite well for me. I detailed the process here.

My site is now live using drupal.

[ax: cherylchase details on cowboyd's method above. moved here from a separate book page (and fixed the HTML).] Senor Dude recommended using a modified Movable Type template to generate all the entries to be migrated as a single file of xml, then cobbling together a quicky parser in perl with XML::SAX to generate mysql insert statements. I had to make a couple of modifications to his code to get the thing working.

[jseng: mt2drupal is another trick you can use to migrate MT to Drupal. It is written in perl as an MT plugin utilizing MT libraries to extract from the database and then feed it into the MySQL database directly. It will import

- all your bloggers
- all your defined categories
- all your entries including body, excerpt, extended (in 4.4.1 & CVS, it is stored as bodyextended and in D4B formatting rules are also preserved)
- all your comments (with anonymous support, in CVS and D4B)
- all incoming trackbacks (stored as comments)
- all your outgoing trackbacks (D4B only)
- all your trackbacks trackers (D4B only)
- keep all your old archives as url_alias, including your RSS feeds so permalink is preserved

[aztek: John Downey has come up with his own conversion script modified from the one in this book. His final site now at <http://www.catdevnull.net> has a view of his experiences starting over again with Drupal.

Extract Movable Type content as xml

1. Install the following as a new Movable Type template called Drupal Convert, with drupal.rdf specified as the Output File.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<items>
<MTEntries lastn="1000" sort_order="ascend">
<item about="{$MTEEntryLink$}>
<title><{$MTEEntryTitle encode_xml="1"}></title>
<description><{$MTEEntryBody encode_xml="1"}></description>
<link><{$MTEEntryLink$}></link>
<subject><{$MTEEntryCategory encode_xml="1"}></subject>
<creator><{$MTEEntryAuthor encode_xml="1"}></creator>
<date><{$MTEEntryDate format="%Y-%m-%dT%H:%M:%S"}><{$MTBlogTimezone$}></date>
</item>
</MTEntries>
</items>
```

2. Save the template, and rebuild. Movable Type will offer to rebuild drupal.rdf for you. The file will contain the last 1000 Movable Type entries, encoded as XML, sorted in ascending date order. This will be helpful if you are turning them into drupal blog entries, because drupal displays blog entries in reverse node id (database insert) order.

Moving your MT styles and templates

Of course you want to make your new Drupal site look and feel as much as possible like your old MT site. This article might help on the way in doing so. Luckily this is not too hard, so non-PHP programmers can re-create the old styles, so that they can be used on the new drupal site. To do this, you do not need to know PHP, but you do need to know at least basic CSS coding.

Since I do not know the way MT templates are built and created, I will stick only to the Drupal part of the story. That's not a too big problem, because, as long as you understand the way drupal uses its themes, you will be able to modify little things so that they /look/ like the MT styles, but do not have to be the same. After all, this is not an article about stealing, its about how to port your own MT creations to drupal. So bear in mind: do not steal!

Drupal knows many methods for theming. It depends on your own preferences what theming method you choose. This, however is far out of the scope of this article, and I stick to the easiest method, in my opinion, PHP template.

The PHP template can be installed as any other theme, read the shipped install text on how to do this. There are numerous articles on drupal.org that explain in detail how to configure PHP template, go look for them yourself and use them to

configure the template. This document is not a tutorial on how to use PHP template, after all.

Creating a new template under PHPtemplate is as easy as copy-pasting one of the folders. Rename it to something you like: MyTemplate for example. The best is to copy the MoveableToDrupal folder and paste it as MyTemplate.

Now some technical blabla on the way PHPtemplate works. Drupal knows themes, just the way most CMS'es do. The file phptemplate.theme is the actual theme. The big difference is that PHPtemplate is not just a theme, like chameleon, or blue robot, but acts more like a layer. It allows you to create templates in the theme. Get it? No?

Alright: Drupal has themes. You can install a theme and the look and feel of your site has changed. But PHPtemplate is not really one of them. It uses some fancy coding to create another templating system. So it is a template in a theme. All the folders inside the PHPtemplate folder are templates.

Got it now?

So inside those sub-folders (for example the folder MyTemplate) there are some files. Some of them are styles, some images and a few are .php files. They are the actual templates. If you know enough PHP just open them and move some of the code around.

But now over to the real stuff: using the style sheet(s) to re-create your MT design.

As said above: you will need CSS skills to do this, if you don't have those, well, post a message on drupal.org or on the support list and tell you've got some money or something else (stories, tutorials, modules, clients with loads of money) lying around you wish to get rid of. And ask if anybody is interested in receiving that money (or that something else) for the simple task of rewriting the MT CSS.

Back to business: drupal can have virtually any HTML DOM, but in general it is kind of similar to that of MT. In PHP template , we have sidebars and a main content. Drupal uses the id .node instead of .blog and it has some more differences, of course.

Some basic changes you should make are:

| MT selector | Drupal PHP template selector |
|---------------|---------------------------------|
| #banner | .header |
| .side | .sidebar-left or .sidebar-right |
| #content | .main-content |
| H3.title | .node H2 (A) |
| .blog | .node |
| .blogbody (P) | .node .content |

Parse xml into sql insert statements

This code was originally published by Senor Dude. But because he didn't publish it in the Drupal Handbook, and because I added code to improve the generation of the teaser (the original code just put the whole body into the teaser), I'm posting it here.

You'll need to have perl of a high enough version to handle the iso-8859-1 encoding (I used 5.8; 5.6 is too old). You'll need to install XML::SAX (easy enough with cpan). This may takes quite a while to run, depending upon what parser it locates.

```
use XML::SAX::Base;
use XML::SAX::ParserFactory;
package Node;
my $teaser_length = 600;
package ConversionFilter;
@ISA = qw(XML::SAX::Base);
my $type = 'blog';
sub characters {
    my ($self,$data) = @_;
    $self->{_characters} .= $data->{Data};
}
sub start_element {
    my ($self, $element) = @_;
    my $tagname = $element->{LocalName};
    my $handle = "start_$tagname";
    if ($self->can($handle)) {
        $self->$handle($element);
    }
    $self->SUPER::start_element($element);
}
sub end_element {
    my ($self, $element) = @_;
    my $tagname = $element->{LocalName};
    my $handle = "end_$tagname";
    if ($self->can($handle)) {
        $self->$handle($element);
    }
    $self->SUPER::end_element();
}
sub start_item {
    my $self = shift;
    $self->{_current_item} = new Node();
}
sub end_item {
    my $self = shift;
    print $self->{_current_item}->insert_statement();
}
```

```
sub start_description {
    my $self = shift;
    $self->clear_characters();
}
sub end_description {
    my $self = shift;
    $self->{_current_item}->{description} =
$self->get_characters();
}
sub start_title {
    my $self = shift;
    $self->clear_characters();
}
sub end_title {
    my $self = shift;
    $self->{_current_item}->{title} =
$self->get_characters();
}
sub start_date {
    my $self = shift;
    $self->clear_characters();
}
sub end_date {
    my $self = shift;
    $self->{_current_item}->{created} =
$self->get_characters();
}
sub clear_characters {
    my $self = shift;
    $self->{_characters} = "";
}
sub get_characters {
    my $self = shift;
    return $self->{_characters};
}
package Node;
sub new {
    my $class = shift;
    return bless {}, $class;
}
# Borrowed from node.module
sub node_teaser {
    my $body = shift;
    my $size = $teaser_length;
    if ($size == 0) {
        return $body;
    }
}
```

```

if (length($body) < $size) {
    return $body;
}
if (my $length = rindex($body, "<br/>", $size)) {
    return substr($body, 0, $length);
}
if (my $length = rindex($body, "<br>", $size)) {
    return substr($body, 0, $length);
}
if (my $length = rindex($body, "</p>", $size)) {
    return substr($body, 0, $length);
}
if (my $length = rindex($body, "\n", $size)) {
    return substr($body, 0, $length);
}
if (my $length = rindex($body, ". ", $size)) {
    return substr($body, 0, $length + 1);
}
if (my $length = rindex($body, "! ", $size)) {
    return substr($body, 0, $length + 1);
}
if (my $length = rindex($body, "? ", $size)) {
    return substr($body, 0, $length + 1);
}
return substr($body, 0, $size);
}

sub insert_statement {
    my $self = shift;
    my $body = mysql_escape($self->{description});
    my $teaser = mysql_escape( node_teaser(
$self->{description} ) );
    return "INSERT INTO node ".
        "(type,title,uid,status,comment, promote,
users, attributes, revisions, created,teaser,body)".
        " VALUES ('$type','".
        mysql_escape($self->{title})."',1,1,2,1,'','','',''.
        "UNIX_TIMESTAMP('.to_mysql_date($self->(created)).')','$teaser','$body');\n";
}

sub mysql_escape {
    my $string = shift;
    $string =~ s/\n//\n/mg;
    $string =~ s/(\'|\")/\\\$1/g; # quote to help syntax
coloring
    return $string;
}

sub to_mysql_date {
    my $string = shift;

```

```

$string =~ s/T/ /;
$string =~ s/\+00:00$/;;
return $string;
}
package main;
my $filename = $ARGV[0];
my $handler = new ConversionFilter();
my $parser = new XML::SAX::ParserFactory->parser(Handler =>
$handler);
$parser->parse_uri($filename);

```

Insert content into Drupal nodes

The final trick is that the insert statements count on mysql's auto_increment feature, but drupal actually sets node ids explicitly. So after you run the generated mysql, you'll need to find the maximum node id that mysql generated for you, and bump the next node id that drupal intends to assign to be larger than that.

```

% /bin/perl ./convert.pl drupal.rdf >mt.sql
% mysql -ppassword drupal select max(nid) from node;
+-----+
| max(nid) |
+-----+
|      83 |
+-----+
1 row in set (0.25 sec)
mysql> select * from sequences;
+-----+-----+
| name      | id   |
+-----+-----+
| users_uid |  8   |
| vocabulary_vid |  2   |
| term_data_tid |  8   |
| node_nid |  6   |
| comments_cid |  3   |
+-----+-----+
5 rows in set (0.00 sec)
mysql> update sequences set node_nid = 84;

```

Setting terms for inserted nodes

you probably want to assign terms to the inserted nodes. You can do this by hand in mysql. The following attaches the term with term id 1 to all the nodes (whose node ids I determined by some characteristic of the nodes themselves, using a select statement).

```
%mysql -ppassword drupal
mysql> insert into term_node select nid, 1 from node where nid >= 6 and nid <= 83;
```

Migrating from PHPNuke

Migrating themes

Just like PHPNuke themes, Drupal uses PHP-based themes mixed with HTML markup.

Both have similar functions for header, footer, box and story (node).

Migrating users

Migrating users:

To migrate users from PHP Nuke to Drupal takes two simple MySQL commands. The following examples are for going from PHP Nuke 5 to Drupal 3.

First, you need to be sure that the 'name' column in the PHP Nuke user table isn't blank. For example, from within MySQL type:

```
update phpnuke.nuke_users set name=uname where name='';
```

Second, copy the valid data from the PHP Nuke user table to the Drupal user table:

```
insert into drupal.users(name,userid,real_email,fake_email,url,bio) select name,uname,email,femail,url,bio from phpnuke.nuke_users;
```

If you find this intimidating, you can try this script which includes more instructions.

Migrating from PostNuke

I've written a MySQL script to migrate from a PostNuke database to a Drupal one.

Currently it migrates Themes, Users, Stories, Comments, Polls and Poll comments. These were the only tables which I was interested in. I wrote it to migrate Puntbarra.COM (the Catalan version of Slashdot) in early September.

It has been a bit complicated given the fact PostNuke database structure is horrible.

Take it from my sandbox.

Configuring mod_rewrite in .htaccess for PN legacy URLs in

I'll be migrating Kairosnews from PostNuke to Drupal CVS this weekend. Since the site has almost 2000 stories, I'm concerned about the fact that any links to them from around the web will go dead. Some consolation is that the .htaccess rules will take those dead links and refer them to the home page instead of 404ing them. Can .htaccess be configured to rewrite the urls? The node ids will be taken from the current story id's, so that part will not be a problem. The current PN default story url looks like this: modules.php?op=modload&name=News&file=article&sid=1972&mode=nested&order=0&thold=0 so it seems like it could be

rewritten to node/view/1972 Is this possible? Admittedly, I know very little about mod_rewrite. Any suggestions would be helpful.

Search engine-friendly migration

The biggest pain for me, having just migrated from Geeklog to Drupal is that my site has a good index in google and other search engines. All those users finding my site on Google are suddenly getting the 404 error page, giving up, and going away when they could dig a bit deeper and find exactly what they were looking for.

So, I harnessed my experience writting HTTP_REFERER logging systems for geeklog to provide myself with a Page with PHP enabled content to parse the Search Engine query from the HTTP_REFERER and do a search on my Drupal site to try and find what the user was actualy looking for. I thought I'd share with the world, as that's the entire point of open source systems like Drupal.

```
<?php
$searchengines = array(
    '^http://www\.google.*$' => 'q',
    '^http://www.google.fi.*$' => 'q',
    '^http://.*search.msn.co.*results.*$' => 'q',
    '^http://.*\mysearch.com/jsp/GGmain.jsp?searchfor=.*$'
=> 'searchfor',
    '^http://search.freeserve.com/.*$' => 'q',
    '^http://aolsearch.aol.co.*$' => 'query',
    '^http://search.yahoo.com.*$' => 'va',
    '^http://search.yahoo.com.*$' => 'p',
    '^http://www.bbc.co.uk/cgi-bin/search/.*' => 'q',
    '^http://www.tiscali.co.uk/search/results.php.*$'
=> 'query',
    '^http://www.altavista.com/web/results.*$' =>
'q',
    '^http://search.hotbot.co.uk/cgi-bin/pursuit.*$'
=> 'query',
    '^http://www.excite.co.uk/search/web/results.*$'
=> 'q',
    '^http://uk.search.yahoo.com/search.*$' => 'p',
    '^http://search.wanadoo.*$' => 'q'
);
$referer = getenv("HTTP_REFERER");
while( list( $regexp, $qsitem ) = each( $searchengines )
)
{
    if( eregi( $regexp, $referer ) )
    {
        echo( t("<br/><h2>Search Engine Detected</h2>It would
appear you arrived here on the wings of a search engine, so, I will
" );
    }
}
```

```

search my local database and show you anything that matches what you
were looking for:<br/>"));
$url = parse_url( $referer );
$querystring = $url['query'];
$querystring = explode( "&", $querystring );
while( list( , $value ) = each( $querystring ) )
{
    $item = explode( "=", $value );
    if( $item[0] == $qsitem )
    {
        if( trim( $item[1] ) != '' )
        {
            $item[1] = urldecode( $item[1] );
            echo ( search_data( $item[1] ) );
        }
    }
}
?>

```

This provides a (partial) list of regular expressions for common search engines, with information as to which query string parameter is the query the user entered. The HTTP_REFERER value (the site the user clicked a link to get to your site) is then examined against this list. When a match is found, a search is done using the standard Drupal search call (search_data). This locates potential matches, and hopefully, keeps the user on your site.

In order to use this, create a new node which allows PHP code. You can call it what you want, and put whatever explanatory text you like on it. You can set whatever path you like. Just drop the above code-clip into place. Then, in Administration -> Settings set the 404 handler to be the path to the new node you created, and voila, if the user arrives from a recognised search engine, their search is performed on your site. It's working nicely for me.

Backups

It is a good idea to backup any data which you would be sad to lose. Or if you would get fired if you lost it.

To backup Drupal data, you need to backup your Drupal database. If you use MySQL, the technique for doing this is here. Somewhat obscured on that page is a suggestion to just copy the right mysql files to another computer (the /data directory). That is a good, easy option.

You might also wish to backup the conf.php file, and any Drupal PHP scripts which you might have customized.

Best practices guidelines

If you are going to invest in the time to setup a CMS, then you should make sure you protect your investment with some simple best practices guidelines. These guidelines are only suggestions on things that need to be considered when managing a web site or CMS. It is up to you to decide what is appropriate for you to implement for your sites.

Accounts and roles

1. Do not use the first user account for day to day stuff on your site. This account should be used for the site setup, major configuration changes and upgrades only.
2. Set up some appropriate roles (do not forget to update these roles as you add new modules). Some role suggestions are 'site admin', 'user admin', 'site contributor'. What roles you need to create will depend on the type of site you have designed. Note that anyone who can administer users can grant themselves additional permissions.
3. Adding new modules often adds additional permissions options. When adding new modules review your access lists to check for appropriate access for your roles.

Backing Up Your Drupal Site

1. Inquire about your ISP or web host's backup policies. Even better, do it yourself periodically. Monthly, weekly, daily or whatever fits your site's needs. Even if you tell yourself, "It's no big deal", it is and you do not want to lose your content.
2. Backing up your MySQL database is covered in section 5.7 of the MySQL handbook. For more information on backing up a MySQL database, read the disaster prevention guidelines. If you are using a different database, then that vendor's documentation should have that information.
3. There are many files that are not part of the database that are important to your site, so periodically backup your entire drupal directory. This will ensure that any uploaded files, pictures, configuration and theme customizations are backed up.
4. Date your backups. Save them in a directory with the date of the backup. You do not want to be trying to figure out when you last backed up your site when you are trying to recover it. Panic is not conducive to a good recovery.
5. One alternative solution to backing up the database of your site, is to install the Database Administration module. It has the option to backup the database tables from within Drupal itself. (Note: Be careful with the permissions.)
6. If you are using third party database management tools such as phpMyAdmin, check out the phpMyAdmin online documentation that can help with creating, managing and backing up your databases.
7. Document your backup and restore procedure so that you can repeat the process easily if necessary. Ideally you want to store the backup and restore

process offline or on a site on a different server than your Drupal site.

File/directory management

Modules that are not part of core may or may not be supported by their contributor for a Drupal version upgrade.

1. Create a sub directory in modules for each contrib module you install. This helps you track contributed modules vs included core modules.
For example: *a module named foo.module would go in a sub directory of modules foo; modules/foo*
2. Rename or remove update.php from the root of your Drupal directory unless you are actually updating your site. There are protections for it in the update script, but why take a chance.
3. If you manage more than one site, consider putting a version.txt file in the root of your drupal directory with the Drupal version, date and modules you are using. If you only manage a few sites you will probably remember them all, but if you set them up for other people, these reminders can help you if you are asked back to do additional work. Also, it can help the next site admin if you move on.

Test Sites

Set up a test site using your live data. You do not want to have to ask in the forums how to save your site. You have worked hard to build it, it would be a shame to lose it.

1. Never do development or testing on your live production site. Drupal is fast and easy to install. Always test on a test site first.
2. Test that your backups work and that you know how to do a restore of your site. The test system can be your local desktop, just edit your conf.php (4.5 or earlier) file or settings.php (4.6) to localhost. You do not want to discover the hard way that you forgot a file or did not know how to do this when your site is down.
3. Test your site upgrade procedure before risking your live site and document the steps you take. Documentation aids repeating the process if necessary. Testing can be done with wget. Read more about how to do that.

Troubleshooting FAQ

Perhaps your question has been asked and answered already. Check this FAQ or perform a search to find an answer. If you don't find an answer here, ask your question in the Support forum.

Installation/configuration

"Headers already sent" error

If you get a "headers already sent" error, there are two likely causes.

If this error is not the first error message on the page, then it is most likely a '**avalance effect**' of previous errors and you may ignore it. Instead, focus on fixing the errors before it. When you fix the first error message(s), the "headers already sent" error(s) will most likely disappear.

However, if you get an error "headers already sent" as the first error, especially when trying to log in and it tells you the error is near the end of a file (check which file "output started at" points to), that probably means that there are **extra spaces or lines** after the closing ?> php tag. Just delete them, and everything should work fine.

The extra whitespace being added probably is caused by a bad unpacking program and / or a windows editor adding it.

"LOCK TABLES sequences WRITE" error

This is in reference to the following kind of error:

```
user    error: Access denied for user:  'YOUR    DATABASE'
USER@localhost' to database 'YOUR DATABASE'
query: LOCK TABLES sequences WRITE in /your
website/database.mysql.inc on line xx
```

When you installed Drupal, you created a database and a database user. This error is caused by that user not having a certain privilege over your database. As far as I can tell, it does not prevent you from posting content, but it does make a lot of ugly errors.

When you come across this problem, first, double check that you have granted all privileges to your user. There are instructions on how to do this in the install file that comes with your Drupal installation.

If you still get this problem, and you are using shared hosting, it is very likely that your host does not usually let users have the kind of permission required. For reference, you may want to read these forum posts. You need to contact your host (e-mail seems better than live support for this) and ask (if you are using mySQL 4) for **global LOCK TABLES privilege** for your user in that database.

Although there are fixes in the forums, these are not recommended. If your host denies your request, they may not be an appropriate place for your Drupal site.

"Method POST is not allowed for the URL /index.htm" error

Solution by: Al

Your Drupal directory contains both an index.html and index.php file. Remove the index.html file or configure your web server to look for index.php first before index.html.

Original posting

.htaccess page forbidden

Solution: Add "FollowSymLinks" to the Options line to the .htaccess file which by default only has "-Indexes". ie use somethings like:

Options FollowSymLinks -Indexes

Original posting

Block referrer spam

In some Content Management Systems, you can configure a block that what links people clicked to come to your site. This is called a referrer sting. Drupal doesnt display this referrer, which is a good thing.

Many porno and online poker sites have robots that are sending "fake" referrer strings, claiming the entered your site via their own site. When you would have a block displaying these referrers, they would get more hits from people clicking these displayed referrer URL's. So it is a good thing that drupal doesnt display these referrers, apart from the referrer logs admin page.

Often, the robots sending these referrers do this via a socalled zombienetwork, thousands and thousands of misconfigured or hacked PC's with are "open proxys". You cant really block this referrer spam, the robots are sending real useragents, are using man URL's and there are so many zombi IP adreses that are chaging fast, that you cant block these.

There is some work underway to block bad behaving robots and users from within drupal. This will probably be part of drupal 4.7. Until that time, the only way to block the referrer spam it by looking at your referrer log and look for often used word in the fake referrer URL's , like "online-poker".

Now you can block these words in your .htaccess file. Say you want to block the referrers "internte-poker" and "viagra" as well as all useragents that contain "looksmart"

First localize your .htaccess. This file is most likely in your drupal document root. You can use vi, pico, notepad or another editor. Now go to the end of the file and just before the last line add:

```
# Block referrer spam
RewriteCond %{HTTP_REFERER} (viagra) [NC,OR
RewriteCond %{HTTP_REFERER} (internet-casino) [NC,OR]
```

```
RewriteCond %{HTTP_USER_AGENT} (Looksmart) [NC]
RewriteRule .* - [F]
```

You dont have to restart your webserver, these settings take place imidiately. When you look at your logging, you will still see the spamming robots with the fake referrer URL's. But you will see that these clients now get a 404 error, this means that they are not allowed to access that (or any other) page.

If they robots that is sending this referrer spam is a "smart", it will know sending the fake URL didnt work. Now it wont stop all the bad guys, they will probably try to send another URL. Or the will go to another site to spam there. But there is a change you will make it a better world. Try it.

E-Mail from Drupal is bouncing or not being sent

If you are not receiving any E-mails from Drupal, or if E-mail sent by Drupal is bouncing, then ensure that the SMTP configuration is set properly in your php.ini.

If you continue to have problems, the use the "user_mail_wrapper" option included with Drupal.

You can now hook up your own custom SMTP library to Drupal instead of using the default PHP mail() function. For more people mail() will work just fine, but for others this is a major problem and it does not work properly. If you just want to get started you will have to download a custom wrapper function from the Drupal contrib repository. If you already have a favorite SMTP function you want to use you will have to create your own wrapper function.

Make an include file that defines a user_mail_wrapper function:
user_mail_wrapper(\$mail, \$subject, \$message, \$header); This function should take the parameters and pass them to the SMTP lib. You will probably have to configure the SMTP lib in some way.

Modify your configuration file (conf.php) to include:

```
$conf["smtp_library"] = "path/to/wrapper.inc";
```

Check out <http://cvs.drupal.org/viewcvs/contributions/tricks/smtp/> for an example.

Originally written by Kjartan on January 9, 2002, with modifications.

Customize smtp.inc from the repository above to ensure that the proper settings for your SMTP server are being used.

How can I administrate my navigation on my drupal site?

A lot of questions come up about how navigation on drupal can be modified, tweaked, altered or any other synonym.

The theme: example, Internet explorer, Netscape, Opera, Lynx has two Navigation block that look like this (assuming you have managed to add link 'Mypage' there by defining a 'page'):

will show:

- * Navigation
- * home
- * news feeds
- * archives
- * blogs
- * books
- * forums
- * My Page
- * polls
- * search

but also:

Navigation

- * create content
- * recent posts
- * news aggregator

Where the second one (create content etc.) changes its name to name of the user after the user logs in.

Themes: marvin and unconed have no generic navigation block but has the same links in the menu on the top of the page.

Other themes do not have any version of generic navigation block.

In the past drupal used to have a , what I call "functional navigation". Each module -each function- could add a link to a general list of links. That list could then be displayed anywhere in drupal. The list has only one level. so sub-elements were not possible.

For most of the CMS powered sites a functional navigation is the best method of navigation: forums, blogs etc all have their own specific content-display and content navigation, based on their function.

But as of drupal 4.3 people started inventing all sorts of navigation modules. These modules would use tabs, blocks, or even hardcoded (D)HTML to make the navigation easy. So in drupal 4.4 there was a general, standard "navigation" block introduced. But this one was not configurable. Only modules could add items in that block.

So as of 4.5RC JonBob together with lots of others came up with a nice menu

system, fully configurable, with permissions and of course multi-levelled (as was the previous too).

Q: I like the first, generic Navigation block but most of the themes do not display it, even after I enable Navigation block in the configuration of blocks.

A: The links list is still present in drupal, even in 4.5. You can print a list of linkes using for example:

```
<?php
$output .= theme("links", link_page(), " " :: " );
?>
```

the list will then be something like blogs :: forum :: mypage :: weblinks

Not all themes use this function. In fact, nowadays only very few do. So you will need to add this manually somwhere.

For example in a custommade sideblock you can say:

```
<?php
return $output .= theme("links", link_page() "<br />" );
?>
```

Please refer to the documentation on drupal.org about printing vs returning in blocks. This is different in some releases of drupal!

Q: What does the Navigation block in block config refer to? (which block displayed above is THE navigation block?)

A: That is the default drupal navigation blok. It offers multi-levelled navigation. It is also a place where some modules place their navigation too (event.module for example). You can disable this block, but you will then not be able to get into your adminstration, other than typing in the urls by hand. So be carefull!

Q: Where is the first (generic) Navigation block defined? What to look for if I want to add it to my theme? I like xtemplate so far so that's where I would like to have the generic navigation block.

A: That is an implementaion of

```
<?php
print $output .= theme("links", link_page() "<br />" ); //Or
something very similar
?>
```

How can I install modules?

Hi, I'm new (very VERY new) to this, and just installed Drupal today. are the extra modules i downloaded to my hard drive installed in the same manner as the regular site? do I need to create individuals files for each module in my public file? (I'm using opensourcehost) Do i need to create a mysql database, etc., for each additional module? Do you know of any

easy-to-understand instructions online? I keep only coming up with partial instructions. Thanks!

How do I unset the clean urls?

After enabling the clean urls in configuration all content is inaccessible, because the system you run drupal on, does not support (all) clean urls.

Clean urls are those fancy looking addresses: instead of `www.server.com/?q=/foo/bar` you see `www.server.com/foo/bar` with clean urls.

Problem is that you cannot set it back, because you cannot browse to the specific page anymore.

There are two solutions:

- The first one is very handy if you have mysql access.
Run the mysql command:

```
UPDATE variable SET value = 's:1:"0"'; WHERE name = 'clean_url';
DELETE FROM cache;
```

- Alternatively, you can modify the appropriate settings.php file: you should add the line

```
$conf['clean_url'] = 0;
```

somewhere in this file.

NOTE: Drupal 4.6 has added the ability to detect whether your site is capable of running with clean url's. If the option to turn them on is greyed out, then you will be unable to activate it until you resolve the underlying issue.

No content on main page for non-admin users

Hi there! First of all thanks for the effort you put in Drupal. It really is a great tool. I just installed 4.1.0 on Linux/Apache1.3 and I'm trying to build my website. I've created an admin user following the installation manual and posted a test story. The story is visible on the main page (`http://localhost/`) only if I'm still logged in as admin. If I click "logout" I get an empty page with the login box on the right and no content. I've checked the status of the item and everything seems OK. Anyone can help? Thanks in advance.

Permission denied in includes/file.inc

This describes an error which may occur with to Drupal 4.5.2. The details for other versions may differ.

`http://www.example.com/?q=admin/settings` might yield error messages like this:

```
warning: mkdir(files): Permission denied in  
/data/www/public/includes/file.inc on line 77.  
warning: mkdir(files/tmp): No such file or directory in  
/usr/data/www/public/includes/file.inc on line 77.
```

This means that Drupal needs write access to create (and later access) the files and files/tmp directories. One way to solve this is to give the webserver write access in the directory. Another common solution seems to be granting everybody write access to the "files" directory. Both solutions has the drawback that somebody else is able to write files into that directory. [if you know a better solution, please mention it here]

Solution 1 (recommended)

In the drupal root directory:

```
mkdir -p files/tmp  
chown -R www files
```

Note: you may need to substitute www with the user id of your webserver process.

Solution 2

In the drupal root directory:

```
mkdir -p files/tmp  
chmod -R 777 files
```

PHP safe mode issue

The error we all hate:

```
warning: Cannot set time limit in safe mode in  
/home/virtual/site12/fst/var/www/html/cron.php on line 11.
```

The reasons:

- 1) Using a host that has Safe Mode enabled
- 2) PEAR is missing

How to Fix it:

- 1) edit your includes/conf.php to show the correct location of PEAR
This line looks like:

```
# If required, update PHP's include path to include your PEAR directory:  
// ini_set("include_path", ".:;/path/to/pear");
```

- 2) Install PEAR and set the above line
- 3) Request your Host to install PEAR
- 4) Find a new host

I hope this helps all those posts regarding the Safe Mode issue.
If you can think of any otehr solutions please add them here.

What is the minimum version of PHP?

Which Version of PHP is required? I am running RedHat 6.2 / Apache 1.3.12 / PHP 3.0.15 / MySQL 3.23.51

Nodes

Can't create static php page

In an attempt to find a temporary work around for kevinlebo's static page problem I created a directory called static in the drupal root directory and in that directory I created a an test.html file. I then created a static php page in drupal that consisted of include() for the html file trying to avoid the large html file from being put in the database where I believe it is getting mangled. However, when I do this I'm getting a parse error in page.module. I only get this when I select php from the list box. This is on Drupal 4.3.0 and any help would be appreciated so kevin can get his site live. Thanks.

PHP content won't parse

Are you getting errors like this?

```
Parse error: parse error in
/home/htdocs/drupal/modules/page.module(105) : eval()'d code on line 1
```

When putting a piece of PHP code in pages/nodes, you must not include the <?php ?> tags around it. If you want to use a mix of PHP and HTML (like in a .php file), start your entry with ?>.

Schedule and expire Nodes

Hello, I have a question about nodes. I didn't see anything about this in the manual. If it's there please excuse the questions. 1)Is it possible to schedule when a node is to appear on the front page? I would like to great a node in advance and schedule when it is to appear. 2)Is it possible to set an expiration date on a node so that it appears static on the front page until a certain date then loses the static flag?
Thanks Joe Cotellese Clearstatic.org

Search

Search index db empty/incomplete

Hi!, I run cron.php, then (trying to fix something) I empty the search index db. Now when I run cron.php again don't process the old nodes (the ones which were in db before empty). How can I do to process those nodes again? Thanks!

Search multibytes language

I am trying to setup a chinese portal website using drupal. In my experimenting I seems to have problem searching chinese. I know the phpbb has no problem searching chinese, so it should not be a php issue. Does anybody have any clue? Thanks.

Polls

Polls allow you to create interactive questionaires that are living parts of your Drupal documents.

Are polls supported in Drupal?

Question: Does Drupal let you create polls?
Yes. Enable the polls module.

Can a user vote more than once in a poll?

Question: Can a user vote more than once in a poll?
In theory but it is actually quite hard. Polls are tied to the user's IP address so he would have to be using at least a different machine.

Miscellaneous

Download offline copy of drupaldocs.org - stuck without net access

I live out in the sticks without broadband Internet access (and I can't get my bloody laptop modem working under linux yet for dial-up access) and need the content of drupaldocs.org. I've downloaded the drupal.org handbook as one massive HTML file by opening the 'printer friendly version' (not ideal, but its a start) but also need the drupaldocs.org content to help me understand the module API better. Ideally, all Drupal documentation should be available for downloading to view offline. I already have such documentation for PHP, MySQL, Apache, Javascript, CSS etc. and it makes life so much easier when you're stuck without net access. Drupal documentation is all that's missing. Ideally the documentation should be downloadable as a series of HTML pages. Better still, there'd be a way to download the code and mysqldump of drupaldocs.org.

How can I change Drupal's character encoding? (UTF-8 and Unicode)

Several people have asked how to specify the character encoding that Drupal uses. The short answer is: you can't, but you don't have to.

Drupal uses UTF-8 for encoding all its data. This is a Unicode encoding, so it can contain data in any language. You no longer need to worry about language specific encodings for your website (such as Big5, GB2312, Windows-1251 or 1256, ...). Also, when Drupal imports

external XML data (such as RSS or XML-RPC), it is automatically converted into UTF-8 (iconv support for PHP will be required for most encodings).

If you really want to change Drupal's encoding, you will experience a lot of troubles, because of the various ways Drupal can receive and send out data (web, e-mail, RSS, XML-RPC, etc).

How do I report a bug in a contributed module?

What is the procedure for reporting bugs in the separately downloadable modules? The bug system does not seem to include these.

How to install a Patch?

I've read the "Diff vs. Patch" thread, but I guess I'm missing something. Perhaps that's because it's about creating a patch, not installing a patch. Could someone please explain:

- Does a patch patch the SQL tables?
- Is "patch" a MySQL command or just a descriptive suffix?
- If this affects the MySQL database, is there a way to install it using phpMyAdmin?
- Does a patch patch the module file?
- If so, how? Is it a php program? Do I ftp it and point to it with a browser?
- (What is it I'm not asking that I should be asking?)

I've searched the posts and cannot find anything that speaks to my dim wits. Any clues would be much appreciated! FWIW, I'm using remote shared hosting (Debian), and, coming from older media, I am comfortable with (x)html and css but still very much learning PHP and know nada re MySQL. Thanks!

Making a custom script work (independently) along with a Drupal setup

This is a doc which deals with running your own (custom) script, along with an existing drupal setup.

Let us assume the file is called test1.php . To get it running, these are the steps which can be followed.

1)Create a new directory (say test), and put the file in it.

Next, copy the .htaccess file from the drupal dir into the dir just created.

Edit the .htaccess file to read

session.save_handler files

This is because Drupal uses a custom session handler, and this needs to be overridden back to default.

2)In case of an error like

Fatal error: session_start(): Failed to initialize storage module. in /home/xyz/public_html/test1.php on line 24

just do a `ini_set('session.save_handler', 'files');` before `session_start()` in your custom script. This creates a new session for the script *only. (or follow step 1)

3)(Long method)

Change the main .htaccess file (in the drupal dir)to read

`session.save_handler files`

NOTE: This however changes the global settings, meaning Drupal **will NOT WORK**, unless the setting is reverted back to user. The custom script however, will work.

HTH,

Viksit Gaur
www.viksit.com
me@viksit.com

Move existing site to new server

I have a drupal site setup on one server. However I want to move it to another server. New domain name, include files in different location, is even in a different country. Is there an easy way to achieve this without having to start from scratch. As in migrate all content and user accounts. Regards, Bawdo2001.

Moving your site to another url

If you want to move your drupal site to a new url it's a good idea to plan this well in advance.

Firstly bear in mind that your new site will not yet be known to search engines, portals etc. You should also take the opportunity to inform people on the old site of the impending changes, giving them the opportunity to update their bookmarks.

But I leave the organizational part up to you.

For the benefit of this exercise I'll assume you have clean urls enabled on both sites, and the new site contains at least the same articles that you had available on your old site.

Moving a site is not difficult, because apache can do most of the work for you using Redirect Directives.

All you have to do is change the .htaccess in your old drupal root. That file should contain only the line:

Redirect permanent / http://yournewdomain.com/

If you are running multiple drupal sites on one domain (e.g. drupal1.yourolddomain.com) you should use:

Redirect permanent / http://drupal1.yournewdomain.com/?q=

Upload the .htaccess and you're done. The old urls will point all users, bots and agents that respect a 301 http header (most do) to your new site. Some browsers will even modify your bookmarks I am told, but mine (firefox) does not.

My URL is wrong in the list of Drupal sites

make sure you call cron.php with a fully qualified domain name (FQDN).

Bad: http://127.0.0.1/cron.php

Good: http://www.mydomain.org/cron.php

Truncated fields/unable to login/PHP 4.2.3 bug

So I finally worked out some issues I had setting up a second mySQL database with my hosting service, and created the initial database for my site. In general, my preferred method of working is to set up a local copy of apache, install the software and misc. add-ons, and then export a copy of the database, which I then import into the database on the server. After doing this, when I tried to log in on the hosted copy, the page refreshed and showed only a portion of the username I'd typed, as well as a shorter password, and the message "I don't recognize that user." After clearing the auto-complete cache in Explorer and simply typing each field out by hand with the same result, I reverted to a "clear" copy of the database with no user info. The new account was created without incident; I chose the user name "Middle Brother", gave it my email address (webmaster@atlasisshrugging.org), and it spat back out "barnerd" as a password. When I tried to log in again, however, the same thing happened. The text fields showed "le Brother" and my password was something like "****". While I know very little about mySQL, I did check the new database and sure enough, the user name had been stored as "le Brother", my email address was "aster@atlasisshrugging.org", etc. I don't have a lot of work invested into the setup and so I'm not going to lose anything if I start over from scratch, which is what I plan to do next. But I'm worried that I'll run into the same problem. Does anyone have any idea what might cause this or how to fix it?

Contributor's guide

The Drupal engine is open source. It is possible for each and every user to become a contributor. The fact remains that most Drupal users, even those skilled in programming arts, have never contributed to Drupal even though most of us had days where we thought to ourselves: "I wish Drupal could do this or that ...". Through this section, we hope to make Drupal more accessible to them.

The guide pages found here are collaborative, but not linked to particular Drupal versions. Because of this, documentation can become out of date. To combat this, we are moving most developer documentation into the Doxygen documentation that is versioned by CVS and generated from the source code. Look there for up-to-date and version-specific information.

- CVS log messages
- Browse CVS repository

Contributing to Drupal

Drupal is a collaborative, community-driven project. This means that the software and its supporting features (documentation, the drupal.org website) are collaboratively produced by users and developers all over the world.

There are several ways to contribute to Drupal:

- Improve or enhance the software
- Provide support and documentation for other users (e.g., by posting additions or updates to the Drupal Handbook or answering requests on user forums or issues).
- Provide financial support to Drupal development.

This section focuses on the first of these three.

Types of Contributions

There are two basic types of contributions you can make to Drupal's code base: (a) "contributed" modules or themes and (b) contributions to the drupal "core".

- "*Contributions*" are the community-produced modules and themes available on the Drupal site. To make a contribution, you need to apply for contributor privileges, produce your contribution, and then notify the contributions manager to request a review of your work before posting. As long as contributions meet some minimal criteria - they do what they claim to and have some demonstrable benefit without unduly replicating already-available functionality - they are approved.

If you have major enhancements you wish to contribute, doing so via a contributed module is in many ways the easiest way to begin. Contributed code has a relatively low set of requirements to meet.

- In contrast, *changes to the Drupal core* are made through a thorough consultative process to ensure the overall integrity of the software.

Changes to the Drupal core are generally of three types:

- *Bug fixes.* These changes respond to identified problems in the existing code.
- *New features.* These changes are enhancements on what is already available.
- *Code maintenance.* These changes are to improve the quality of the code or bring it up to date with changes elsewhere in Drupal. This can include bringing code in line with coding standards, improving efficiency (e.g., eliminating unneeded database queries), introducing or improving in-line comments, and doing upgrades for compliance with a new release version.

While you can create your own issues, you can also begin by simply taking on existing tasks on the task list.

Task list

The Drupal bug database contains many issues classified as "bite-sized" tasks -- tasks that are well-defined and self-contained, and thus suitable for a volunteer looking to get involved with the project. You don't need broad or detailed knowledge of Drupal's design to take on one of these, just a pretty good idea of how things generally work, and familiarity with the coding guidelines. Each task is something a volunteer could pick off in a spare evening or two.

If you start one of these, please notify the other developers by mailing drupal-devel@drupal.org (of course, you should be subscribed to that list). If you have questions as you go, ask the dev list or update the task (updates are sent to the list automatically). Send the patch to the list when ready.

Bug reports

If you found a bug, send us the bug report and we will fix it provided you include enough diagnostic information for us to go on. Your bug reports play an essential role in making Drupal reliable.

Bug reports can be posted in connection with any project hosted on drupal.org. You can submit a new bug via the submit issue form. Provide a sensible title for the bug, and choose the project you think you have found the bug in. After previewing the submission, you will need to choose a related component and you

will be able to provide more details about the bug, including the description of the problem itself. Please include any error messages you received and a detailed description of what you were doing at the time.

Note that you don't have to be logged in nor a member of drupal.org to submit bugs.

The first thing we will do when you report a bug is tell you to upgrade to the newest version of Drupal, and then see if the problem reproduces. So you'll probably save us both time if you upgrade and test with the latest version before sending in a bug report.

How to report bugs effectively

Summary

- The first aim of a bug report is to let the programmer see the failure with their own eyes. If you can't be with them to make it fail in front of them, give them detailed instructions so that they can make it fail for themselves.
- In case the first aim doesn't succeed, and the programmer can't see it failing themselves, the second aim of a bug report is to describe what went wrong. Describe everything in detail. State what you saw, and also state what you expected to see. Write down the error messages, especially if they have numbers in.
- When your computer does something unexpected, freeze. Do nothing until you're calm, and don't do anything that you think might be dangerous.
- By all means try to diagnose the fault yourself if you think you can, but if you do, you should still report the symptoms as well.
- Be ready to provide extra information if the programmer needs it. If they didn't need it, they wouldn't be asking for it. They aren't being deliberately awkward. Have version numbers at your fingertips, because they will probably be needed.
- Write clearly. Say what you mean, and make sure it can't be misinterpreted.
- Above all, be precise. Programmers like precision.

Introduction

Anybody who has written software for public use will probably have received at least one bad bug report. Reports that say nothing ("It doesn't work!"); reports that make no sense; reports that don't give enough information; reports that give wrong information. Reports of problems that turn out to be user error; reports of problems that turn out to be the fault of somebody else's program; reports of problems that turn out to be network failures.

There's a reason why technical support is seen as a horrible job to be in, and that reason is bad bug reports. However, not all bug reports are unpleasant: I maintain free software, when I'm not earning my living, and sometimes I receive wonderfully clear, helpful, informative bug reports.

In this essay I'll try to state clearly what makes a good bug report. Ideally I would like everybody in the world to read this essay before reporting any bugs to anybody. Certainly I would like everybody who reports bugs to me to have read it.

In a nutshell, the aim of a bug report is to enable the programmer to see the program failing in front of them. You can either show them in person, or give them careful and detailed instructions on how to make it fail. If they can make it fail, they will try to gather extra information until they know the cause. If they can't make it fail, they will have to ask you to gather that information for them.

In bug reports, try to make very clear what are actual facts ("I was at the computer and this happened") and what are speculations ("I think the problem might be this"). Leave out speculations if you want to, but don't leave out facts.

When you report a bug, you are doing so because you want the bug fixed. There is no point in swearing at the programmer or being deliberately unhelpful: it may be their fault and your problem, and you might be right to be angry with them, but the bug will get fixed faster if you help them by supplying all the information they need. Remember also that if the program is free, then the author is providing it out of kindness, so if too many people are rude to them then they may stop feeling kind.

"It doesn't work."

Give the programmer some credit for basic intelligence: if the program really didn't work at all, they would probably have noticed. Since they haven't noticed, it must be working for them. Therefore, either you are doing something differently from them, or your environment is different from theirs. They need information; providing this information is the purpose of a bug report. More information is almost always better than less.

Many programs, particularly free ones, publish their list of known bugs. If you can find a list of known bugs, it's worth reading it to see if the bug you've just found is already known or not. If it's already known, it probably isn't worth reporting again, but if you think you have more information than the report in the bug list, you might want to contact the programmer anyway. They might be able to fix the bug more easily if you can give them information they didn't already have.

This essay is full of guidelines. None of them is an absolute rule. Particular programmers have particular ways they like bugs to be reported. If the program comes with its own set of bug-reporting guidelines, read them. If the guidelines that come with the program contradict the guidelines in this essay, follow the ones that come with the program!

If you are not reporting a bug but just asking for help using the program, you should state where you have already looked for the answer to your question. ("I looked in chapter 4 and section 5.2 but couldn't find anything that told me if this is possible.") This will let the programmer know where people will expect to find the answer, so they can make the documentation easier to use.

"Show me"

One of the very best ways you can report a bug is by showing it to the programmer. Stand them in front of your computer, fire up their software, and demonstrate the thing that goes wrong. Let them watch you start the machine, watch you run the software, watch

how you interact with the software, and watch what the software does in response to your inputs.

They know that software like the back of their hand. They know which parts they trust, and they know which parts are likely to have faults. They know intuitively what to watch for. By the time the software does something obviously wrong, they may well have already noticed something subtly wrong earlier which might give them a clue. They can observe everything the computer does during the test run, and they can pick out the important bits for themselves.

This may not be enough. They may decide they need more information, and ask you to show them the same thing again. They may ask you to talk them through the procedure, so that they can reproduce the bug for themselves as many times as they want. They might try varying the procedure a few times, to see whether the problem occurs in only one case or in a family of related cases. If you're unlucky, they may need to sit down for a couple of hours with a set of development tools and really start investigating. But the most important thing is to have the programmer looking at the computer when it goes wrong. Once they can see the problem happening, they can usually take it from there and start trying to fix it.

"Show me how to show myself"

This is the era of the Internet. This is the era of worldwide communication. This is the era in which I can send my software to somebody in Russia at the touch of a button, and he can send me comments about it just as easily. But if he has a problem with my program, he can't have me standing in front of it while it fails. "Show me" is good when you can, but often you can't.

If you have to report a bug to a programmer who can't be present in person, the aim of the exercise is to enable them to reproduce the problem. You want the programmer to run their own copy of the program, do the same things to it, and make it fail in the same way. When they can see the problem happening in front of their eyes, then they can deal with it.

So tell them exactly what you did. If it's a graphical program, tell them which buttons you pressed and what order you pressed them in. If it's a program you run by typing a command, show them precisely what command you typed. Wherever possible, you should provide a verbatim transcript of the session, showing what commands you typed and what the computer output in response.

Give the programmer all the input you can think of. If the program reads from a file, you will probably need to send a copy of the file. If the program talks to another computer over a network, you probably can't send a copy of that computer, but you can at least say what kind of computer it is, and (if you can) what software is running on it.

"Works for me, so what goes wrong?"

If you give the programmer a long list of inputs and actions, and they fire up their own copy of the program and nothing goes wrong, then you haven't given them enough information. Possibly the fault doesn't show up on every computer; your system and theirs may differ in some way. Possibly you have misunderstood what the program is supposed to do, and you are both looking at exactly the same display but you think it's wrong and they know it's right.

So also describe what happened. Tell them exactly what you saw. Tell them why you think what you saw is wrong; better still, tell them exactly what you expected to see. If you say "and then it went wrong", you have left out some very important information.

If you saw error messages then tell the programmer, carefully and precisely, what they were. They are important! At this stage, the programmer is not trying to fix the problem: they're just trying to find it. They need to know what has gone wrong, and those error messages are the computer's best effort to tell you that. Write the errors down if you have no other easy way to remember them, but it's not worth reporting that the program generated an error unless you can also report what the error message was.

In particular, if the error message has numbers in it, do let the programmer have those numbers. Just because you can't see any meaning in them doesn't mean there isn't any. Numbers contain all kinds of information that can be read by programmers, and they are likely to contain vital clues. Numbers in error messages are there because the computer is too confused to report the error in words, but is doing the best it can to get the important information to you somehow.

At this stage, the programmer is effectively doing detective work. They don't know what's happened, and they can't get close enough to watch it happening for themselves, so they are searching for clues that might give it away. Error messages, incomprehensible strings of numbers, and even unexplained delays are all just as important as fingerprints at the scene of a crime. Keep them!

If you are using Unix, the program may have produced a core dump. Core dumps are a particularly good source of clues, so don't throw them away. On the other hand, most programmers don't like to receive huge core files by e-mail without warning, so ask before mailing one to anybody. Also, be aware that the core file contains a record of the complete state of the program: any "secrets" involved (maybe the program was handling a personal message, or dealing with confidential data) may be contained in the core file.

"So then, I tried..."

There are a lot of things you might do when an error or bug comes up. Many of them make the problem worse. A friend of mine at school deleted all her Word documents by mistake, and before calling in any expert help, she tried reinstalling Word, and then she tried running Defrag. Neither of these helped recover her files, and between them they scrambled her disk to the extent that no Undelete program in the world would have been able to recover anything. If she'd only left it alone, she might have had a chance.

Users like this are like a mongoose backed into a corner: with its back to the wall and seeing certain death staring it in the face, it attacks frantically, because doing something has to be better than doing nothing. This is not well adapted to the type of problems computers produce.

Instead of being a mongoose, be an antelope. When an antelope is confronted with something unexpected or frightening, it freezes. It stays absolutely still and tries not to attract any attention, while it stops and thinks and works out the best thing to do. (If antelopes had a technical support line, it would be telephoning it at this point.) Then, once it has decided what the safest thing to do is, it does it.

When something goes wrong, immediately stop doing anything. Don't touch any buttons at all. Look at the screen and notice everything out of the ordinary, and remember it or write it down. Then perhaps start cautiously pressing "OK" or "Cancel", whichever seems safest. Try to develop a reflex reaction - if a computer does anything unexpected, freeze.

If you manage to get out of the problem, whether by closing down the affected program or by rebooting the computer, a good thing to do is to try to make it happen again. Programmers like problems that they can reproduce more than once. Happy programmers fix bugs faster and more efficiently.

"I think the tachyon modulation must be wrongly polarised."

It isn't only non-programmers who produce bad bug reports. Some of the worst bug reports I've ever seen come from programmers, and even from good programmers.

I worked with another programmer once, who kept finding bugs in his own code and trying to fix them. Every so often he'd hit a bug he couldn't solve, and he'd call me over to help. "What's gone wrong?" I'd ask. He would reply by telling me his current opinion of what needed to be fixed.

This worked fine when his current opinion was right. It meant he'd already done half the work and we were able to finish the job together. It was efficient and useful.

But quite often he was wrong. We would work for some time trying to figure out why some particular part of the program was producing incorrect data, and eventually we would discover that it wasn't, that we'd been investigating a perfectly good piece of code for half an hour, and that the actual problem was somewhere else.

I'm sure he wouldn't do that to a doctor. "Doctor, I need a prescription for Hydroxyoxydyne." People know not to say that to a doctor: you describe the symptoms, the actual discomforts and aches and pains and rashes and fevers, and you let the doctor do the diagnosis of what the problem is and what to do about it. Otherwise the doctor dismisses you as a hypochondriac or crackpot, and quite rightly so.

It's the same with programmers. Providing your own diagnosis might be helpful sometimes, but always state the symptoms. The diagnosis is an optional extra, and not an alternative to giving the symptoms. Equally, sending a modification to the code to fix the problem is a useful addition to a bug report but not an adequate substitute for one.

If a programmer asks you for extra information, don't make it up! Somebody reported a bug to me once, and I asked him to try a command that I knew wouldn't work. The reason I asked him to try it was that I wanted to know which of two different error messages it would give. Knowing which error message came back would give a vital clue. But he didn't actually try it - he just mailed me back and said "No, that won't work". It took me some time to persuade him to try it for real.

Using your intelligence to help the programmer is fine. Even if your deductions are wrong, the programmer should be grateful that you at least tried to make their life easier. But report the symptoms as well, or you may well make their life much more difficult instead.

"That's funny, it did it a moment ago."

Say "intermittent fault" to any programmer and watch their face fall. The easy problems are the ones where performing a simple sequence of actions will cause the failure to occur. The programmer can then repeat those actions under closely observed test conditions and watch what happens in great detail. Too many problems simply don't work that way: there will be programs that fail once a week, or fail once in a blue moon, or never fail when you try them in front of the programmer but always fail when you have a deadline coming up.

Most intermittent faults are not truly intermittent. Most of them have some logic somewhere. Some might occur when the machine is running out of memory, some might occur when another program tries to modify a critical file at the wrong moment, and some might occur only in the first half of every hour! (I've actually seen one of these.)

Also, if you can reproduce the bug but the programmer can't, it could very well be that their computer and your computer are different in some way and this difference is causing the problem. I had a program once whose window curled up into a little ball in the top left corner of the screen, and sat there and sulked. But it only did it on 800x600 screens; it was fine on my 1024x768 monitor.

The programmer will want to know anything you can find out about the problem. Try it on another machine, perhaps. Try it twice or three times and see how often it fails. If it goes wrong when you're doing serious work but not when you're trying to demonstrate it, it might be long running times or large files that make it fall over. Try to remember as much detail as you can about what you were doing to it when it did fall over, and if you see any patterns, mention them. Anything you can provide has to be some help. Even if it's only probabilistic (such as "it tends to crash more often when Emacs is running"), it might not provide direct clues to the cause of the problem, but it might help the programmer reproduce it.

Most importantly, the programmer will want to be sure of whether they're dealing with a true intermittent fault or a machine-specific fault. They will want to know lots of details about your computer, so they can work out how it differs from theirs. A lot of these details will depend on the particular program, but one thing you should definitely be ready to provide is version numbers. The version number of the program itself, and the version number of the operating system, and probably the version numbers of any other programs that are involved in the problem.

"So I loaded the disk on to my Windows . . ."

Writing clearly is essential in a bug report. If the programmer can't tell what you meant, you might as well not have said anything.

I get bug reports from all around the world. Many of them are from non-native English speakers, and a lot of those apologise for their poor English. In general, the bug reports with apologies for their poor English are actually very clear and useful. All the most unclear reports come from native English speakers who assume that I will understand them even if they don't make any effort to be clear or precise.

- Be specific. If you can do the same thing two different ways, state which one you used. "I selected Load" might mean "I clicked on Load" or "I pressed Alt-L". Say which you did. Sometimes it matters.
- Be verbose. Give more information rather than less. If you say too much, the

programmer can ignore some of it. If you say too little, they have to come back and ask more questions. One bug report I received was a single sentence; every time I asked for more information, the reporter would reply with another single sentence. It took me several weeks to get a useful amount of information, because it turned up one short sentence at a time.

- Be careful of pronouns. Don't use words like "it", or references like "the window", when it's unclear what they mean. Consider this: "I started FooApp. It put up a warning window. I tried to close it and it crashed." It isn't clear what the user tried to close. Did they try to close the warning window, or the whole of FooApp? It makes a difference. Instead, you could say "I started FooApp, which put up a warning window. I tried to close the warning window, and FooApp crashed." This is longer and more repetitive, but also clearer and less easy to misunderstand.
- Read what you wrote. Read the report back to yourself, and see if you think it's clear. If you have listed a sequence of actions which should produce the failure, try following them yourself, to see if you missed a step.

Feature suggestions

How many times you have dreamed "Gee...I wish Drupal could do that" or "I like the xxx feature, but it should work better". If you want to improve Drupal, send us your wishes as a feature suggestions. Your suggestions play an essential role in making Drupal more usable and feature-rich.

The core features provided by Drupal are listed on the features page. You can submit a feature request by creating a new issue connected to the component the feature is related to. Please note that there is a Drupal contributed module named 'Features' which is used on the feature page mentioned above. Every module has a feature request subcategory, and thus **the 'Feature' module is not the appropriate place** to submit feature requests. To properly file a feature request, first choose the project it is related to and then after hitting preview set the other related options. You will be able to categorize the issue as a feature request with the Issue Information / Category dropdown.

Note that you don't have to be logged in nor to be a member of drupal.org to suggest features.

Patches

Patches are a way to distribute relatively small changes to code. They are the preferred way to contribute bug fixes and other proposed changes to Drupal's codebase.

Diff and patch

Diff and patch are two complementary tools for recording and applying changes between two sets of files.

We use them for content control even though we distribute our code via CVS. Why? Because diff and patch provide an immense amount of control. Patches can be submitted via e-mail and in plain text; maintainers can read and judge the patch before it ever gets near a tree. It allows maintainers to look at changes easily without blindly integrating them.

Diff is the first command in the set. It has the simple purpose to create a file called a *patch* or a *diff* which contains the differences between two text files or two groups of text files. Diff can write into different formats, although the unified difference format is preferred. The patches this command generates are much easier to distribute and allow maintainers to see quickly and easily what changed and to make a judgement.

Patch is diff's complement and takes a patch file generated by diff and applies it against a file or a group of files.

The actual usage of diff and patch is not complicated.

At its simplest, a diff command for comparing two files would be:

```
diff old.txt new.txt > oldnew.patch
```

For drupal, we prefer patches in unified format, so we add -u to the command line:

```
diff -u old.txt new.txt > oldnew.patch
```

It is helpful to keep a reference in the patch file to which function was patched, so the following form of the command is often used. For example, if you have made a change in foo.module, to create a patch against the CVS tree:

```
cvs diff -u -F ^function foo.module > foo.patch
```

Or if you had downloaded Drupal instead of checking it out from CVS and were creating a patch against a local copy of foo.module:

```
diff -u -F ^function foo.module newfoo.module > foo.patch
```

Generally, however, a comparison of two source trees is often desired. A possible command to do so is:

```
diff -ruN old new > tree.diff
```

Once a patch is generated, the process of patching the file is even simpler. Based on our examples above, we could do:

```
patch < oldnew.patch
```

Or if you want to patch an entire directory, you should use:

```
patch -p0 -u < tree.diff
```

To unapply the patch, use:

```
patch -p0 -R < tree.diff
```

Diff on Windows

(against a cvs source with the cvs.exe built-in diff. do diff local files, you need a windows diff program, command line or visual)

Generic:

- find the cvs.exe of your cvs package (WinCVS, TortoiseCVS, cygwin, ...) and make sure it is in your PATH
- cd to your drupal root dir
- cvs diff -u [[-r rev1|-D date1] [-r rev2|-D date2]] [file_to_diff] [> file_to_diff.patch]
 - -u: unified format
 - -r: revision(s) to diff
 - no -r: compare the working file with the revision it was based on
 - one -r: compare that revision with your current working file
 - two -r: compare those two revisions
 - -D: use a date_spec to specify revisions. examples: "1972-09-24 20:05", "24 Sep 1972 20:05".
 - file_to_diff: path to the file or directory you want to diff. if you specify a directory, the output will include the diff of all differing files in this directory and all subdirectories.
 - > file_to_diff.patch: creates a patch - saves the diff in file_to_diff.patch instead of outputting it on stdout. if you send a patch, make sure it has the proper line endings
 - see the CVS manual for a complete list of and additional options

via WinCVS GUI

Just select the file you edited and right-mouse-click > "diff selection" (or press the "diff selected"-icon on the toolbar, or do Menubar > "Query" > "diff selection"). This brings up a "Diff settings" dialog box that offers some limited options as "revisions to diff" and "ignore whitespace/case" [*update 2003-Feb-07: starting with WinCvs 1.3b11, "Full diff options [are] available from the diff dialog"*]. The resulting diff is output to the WinCVS-Console and can be copied and pasted.

via WinCVS/TortoiseCVS external diff

- WinCVS: Menubar > "Admin" > "Preferences" > "WinCVS" > "External diff program ". This program will be invoked by the "Diff selection" when "Use the external diff" is checked.
- TortoiseCVS: CVS > "Preferences" > "External diff application". This program will be invoked by "CVS Diff ..."

Some external visual diff programs for Windows:

- Araxis Merge (commercial)
- ExamDiff

- CSDiff
- for those who can live w/ java: Guiffy (commercial)
- WinMerge
- you may find more here

Notes:

- While these programs do a nice job in showing file differences visually, side by side, *none of them* (as I can tell) allows to actually *save* the difference in unified format (most allow to save a *standard* diff, though) - **update:** TortoiseCVS lets you save patches. It does unified format by default. See its Make Patch option. Note that this 'Make Patch' option can make recursive patches when applied to directories.
- You *cannot* specify the "-u" in the External diff preferences (eg "diff -u") as this will result in "Unable to open 'diff -u'" (The system cannot find the file specified.)". A workaround for this is to, in the preferences, specify a batch-file that calls the external diff with the -u option. Another workaround is meta-diff, which allows for launching of special diff programs for certain file types.)

line endings: an issue with using diff on windows is that generated patches have windows line endings, which makes them impossible to apply on unix boxes [1][2]. unfortunately, there seems to be no way to convince "cvs diff" to output unix line endings*. so the only way for making a proper patch on windows that I see is to convert / filter the output from "cvs diff" to unix line endings:

- filter: pipe "cvs diff"s output through some dos2unix tool (like the one from Robert B. Clark, or like cygwin's dos2unix / d2u):

```
cvs diff [options] file_to_diff | unix2dos -u > file_to_diff.patch
```

- convert: save "cvs diff"s output to a file:

```
cvs diff [options] file_to_diff > file_to_diff.patch
```

and manually convert `file_to_diff.patch` to unix line endings. every developer's editor should be capable of this; besides, there are many dos2unix versions that operate on files.

Patch on Windows

I haven't found any Windows-GUI for patch, so the only choice is a Windows port of the Unix command-line tool. If you know of a Windows GUI for patch, please let me know

- Cygwin - a UNIX environment for Windows, including many standard UNIX-tools (including diff and patch)
- pre-compiled binaries, available from various places. Some I've found:
 - <http://www.squirrel.nl/people/jvromans/tpj0403-0016b.html>

- <http://www.gnu.org/software/emacs/windows/faq11.html#patch>

Note:

I found many of the precompiled binaries have problems with pathnames etc. and do not work properly. So I would recommend installing cygwin - it takes a while, but after that, you have a nice Unix environment that works.

* and i tried a lot: checking out all files with unix line endings, various -kb options, external diffs, patched cvs versions ... nothing. for a discussion of this, check CVS and binary files

You can try also integrated diff/patch packages like GNU diffutils for Windows or get the GNU utilities for Win 32.

Creating and submitting patches

The process of submitting a patch can seem daunting at first. This text is a collection of suggestions which can greatly increase the chances of your change being accepted.

The easiest way to get set up for making and sending patches is to get CVS working. Then you can just type: `cvs diff -u -F^f [file to patch]` to generate a patch. To output it to a file, go: `cvs diff -u -F^f [file to patch] > [outfile]`

Coding style:

If your code deviates too much from the Code Conventions, it is more likely to be rejected without further review and without comment.

diff -u:

Use `diff -u` or `diff -urN` to create patches: when creating your patch, make sure to create it in "unified diff" format, as supplied by the `-u` argument to `diff`. Patches should be based in the root source directory, not in any lower subdirectory. Make sure to create patches against a "vanilla", or unmodified source tree.

diff -F^f:

Use the additional `-F^f` argument to `diff` to create patches that are easier to read. `-F^f` tells `diff` to include the last matching line in the header of the created patch. This will be the last function definition if the files adhere to the Drupal Code Conventions.

Describe your changes:

Describe the technical detail of the change(s) your patch includes and try to be as specific as possible. Note that we prefer technical reasoning above marketing: give us clear reasons why "this way" is good. Justify your changes and try to carry enough weight. It is important to note the version to which this patch applies.

Separate your changes:

Separate each logical change into its own patch. For example, if your changes include both bug fixes and performance enhancements, separate those changes into two or more patches. If your changes include an API update, and a new module which uses that new API, separate those into two patches.

Verifying your patch

The CVS review team is overloaded reviewing patch submissions. Please make their lives easier by assuring the following:

- Test your code!
- Make sure your code is clean and secure. If your patch is just a quick hack, then don't set your issue to **Patch** status.
- Patch against HEAD. If you only have a patch against a prior revision, then don't assign your issue to **Patch** status

Submitting your patch:

Patches should be submitted via the issue tracker. Create a bug report or feature request, attach your patch using the file upload form and set the issue's status to *patch*. Setting the status to patch is important as it adds the patch to the patch queue.

Rules of reviewing patches

1. Do review the code not the person.
2. Do not take a review personally. If you get a bad review deal with it and make your patch better. It is a learning experience.
3. Do help to review other peoples code as it will make your own code better. It will make your more critical and likely to spot your own mistakes. Might also teach you a trick or two you didn't know about.
4. Do not feel obligated to review others code even if people review your code. (It comes highly recommended.)
5. Do give friendly suggestions on how a person can improve their code.
6. Do not demand that your code gets reviewed. Your time will come.
7. Do remind people nicely that it would be nice if someone reviewed your code, but only once a week.
8. Do this to get a good review:
 9.
 - Do make sure the code actually works. Working code is a big plus.
 - Do make sure the patch is current with Drupal CVS. Feel free to refuse to review patches that don't apply nicely to Drupal CVS.
 - Do look at the code and make sure it follows the Drupal coding standards.
 - Do make sure the code uses available support functions and doesn't re-invent the wheel.
 - Do write documentation both in the code and for the users.
10. Do this when reviewing:
 11.
 - Do make sure the patch does everything in item 8.

- Do comment on the general coding style.
 - Do comment on the user interface.
 - Do make suggestions on how to improve the patch.
 - Do give your vote (+1/-1) as to whether this should be included in Drupal.
12. Just do it.

The revision process

Changes to the Drupal core are usually made after consideration, planning, and consultation. They are also made on a priority basis--fixes come before additions, and changes for which there is a high demand come before proposals that have gone relatively unnoticed. Any potential change has to be considered not only on its own merits but in relation to the aims and principles of the project as a whole.

The particular stages that a new feature goes through vary, but a typical cycle for a significant change might include:

- General discussion of the idea, for example through a posting in a drupal.org forum. This can be a chance to gauge support and interest, scope the issue, and get some direction and suggestions on approaches to take. If you're considering substantive changes, starting out at the discussion level - rather than jumping straight into code changes - can save you a lot of time.
- Posting an issue through the drupal.org project system.
- Discussion raising issues on the proposed direction or solution, which may include a real-time meeting through IRC.
Individual Drupal community members may vote for (+1) or against (-1) the change. While informal, this voting system can help quantify support.
- Producing a patch with specific proposed code changes.
- Review of the changes and further discussion.
- Revisions to address issues.
- Possible application of the patch.

The process of discussion and revision might be repeated several times to encompass diverse input. At any point in the process, the proposal might be:

- Shelved as impractical or inappropriate.
- Put off until other logically prior decisions are made.
- Rolled into another related initiative.
- Superceded by another change.

If you submit suggestions that don't end up being adopted, please don't be discouraged! It doesn't mean that your ideas weren't good--just that they didn't end up finding a place. The discussion itself may have beneficial outcomes. It's all part of collaboratively building a quality open source project.

Criteria for evaluating proposed changes

The following criteria are used by core developers in reviewing and approving proposed changes:

- *The changes support and enhance Drupal project aims.*
- *The proposed changes are current.* Especially for new features, priority is usually given to development for the "HEAD" (the most recent development version of the code, also referred to as the CVS version) as opposed to released versions. There may have been significant changes since the last release, so developing for the CVS version means that
- *The proposed change doesn't raise any significant issues or risks.* Specifically, issues that have been raised in the review process have been satisfactorily addressed.
- *The changes are well coded.* At a minimum, this means coding in accordance with the Drupal coding standards. But it also means that the coding is intelligent and compact. Elegant solutions will have greater support than cumbersome ones that accomplish the same result.
- *There is demonstrated demand and support for the change.* Demand is indicated by, e.g., comments on the drupal.org issues system or comments in forums or the drupal-dev email list.
- *The change will be used by a significant portion of the installed Drupal base as opposed being relevant only to a small subset of Drupal users.*
- *The benefits of the change justifies additional code and resource demands.* Every addition to the code base increases the quantity of code that must be actively maintained (e.g., updated to reflect new design changes or documentation approaches). Also, added code increases the overall Drupal footprint through, e.g., added procedure calls or database queries. Benefits of a change must outweigh these costs.

Maintaining a project on drupal.org

Each *drupal.org project* (a contributed theme, module or translation) needs to be maintained in the contributions repository. Before creating a project page on drupal.org, apply for a CVS account and commit your project to the repository. If you are not using the drupal.org infrastructure, you can't setup a project page on drupal.org nor can you offer your module for download at drupal.org.

To get your project listed on drupal.org after it has been committed to CVS, fill in the form at http://drupal.org/node/add/project_project/. Make sure that the 'Short project name' matches the directory name in the CVS repository. For example, the *contributions/modules/my_module* module has the short name *my_module*.

Note that the newly created project will not be instantly available as it will need to be approved by one of the administrators. After that, it will appear soon after you committed some code/updates to the contributions repository. Once the project page became available, people will be able to file bugs against your project, add tasks or request new features. Your project will also become available for download.

Downloads and packaging

As soon your project page has been activated and assuming it is properly configured, drupal.org will automatically package your project and make it available for download. Projects are packaged once or twice a day so your project will not be available instantly.

Managing releases

Releases are handled using CVS branches. By default, only the *CVS HEAD version* (development version) of your project is packaged and offered for download.

However, if you branch your project using the *DRUPAL-4-5* branch name, drupal.org will package the *Drupal 4.5 compatible release* of your project. For this to work, you must use the correct branch names. A list of valid branch names can be found in the contributions repository's FAQ.txt.

As projects are only packaged once or twice a day, it might take up to 24 hours for new releases to become available on the website or for updates to propagate to the downloads.

If you found a bug that needs to be fixed in several releases of your project, make sure to commit the fix to the different branches unless you are no longer maintaining certain releases of your project.

Branching and releases are restricted to the modules, themes, theme-engines and translations directories in the contributions repository. Personal sandboxes in the sandbox directory can't be branched, won't be packaged and can't get a project page on drupal.org.

Orphaned projects

If you are no longer capable of maintaining your project, please add a note to your project page and ask in the forums whether someone is willing to take over maintenance. Proper communication is key so make sure to mark your project as orphaned. If you found a new maintainer or if you are willing to maintain an orphaned project, get in touch with a site maintainer so we can transfer maintainership.

Tips for contributing to the core

The following tips might improve the chances of your contributions being accepted:

- Take a step back and objectively evaluate whether the changes are appropriate for the Drupal core. Ask yourself:
 - Is the feature already implemented? Search the forums and issue tracker.
 - Could the feature be implemented as a contributed module rather than a patch to the core?
 - Will the change benefit a substantial portion of the Drupal install base?
 - Is the change sufficiently general for others to build upon cleanly?
- Be explanatory, provide descriptions and illustrations, make a good case. Don't count on others downloading, installing, and testing your changes. Rather, show them in a nutshell what your changes would mean. Anticipate and address questions or concerns. If appropriate, provide screenshots.
- Be friendly and respectful. Acknowledge the effort others put in.
- Be open to suggestions and to other ways of accomplishing what you're aiming for.
- Be persistent. If you don't get any response right away, don't necessarily give up. If you're still convinced your idea has merit, find another way to present it.
- Respond, in a timely way, to suggestions, requests, or issues raised. Revise your work accordingly.
- If some time has gone by, update your changes to work with the current CVS version.

Mailing lists

Newsletter

A read-only mailing list used for sending out the Drupal newsletter.

[view archive](#) · [search archive](#) · [mailman page](#)

Drupal-support

If you need help with installing, running or anything Drupal related this is the list to post your questions.

[view archive](#) · [search archive](#) · [mailman page](#)

Drupal-devel

This list is for those who want to either take part or just observe Drupal development.

[view archive](#) · [search archive](#) · [mailman page](#)

Drupal-docs

The place for non-programmers that want to contribute and work on documentation.

[view archive](#) · [search archive](#) · [mailman page](#)

Drupal-cvs

All CVS commits are posted to this list, a daily digest is also posted to drupal-devel though.

[view archive](#) · [search archive](#) · [mailman page](#)

Infrastructure

A mailing list for those maintaining the Drupal infrastructure, most notably the drupal.org website.

[view archive](#) · [search archive](#) · [mailman page](#)

Subscribe

Mail address:

Lists:

- newsletter
- drupal-support
- drupal-devel
- drupal-docs
- drupal-cvs
- infrastructure

Mailing of project issues

Every project issue for Drupal *with patch status* is emailed to the drupal-devel mailing list when updated. These are mailed to promote peer review of code potentially going into Drupal. Other issues are not emailed because it would make the mailing

list less useful as email volume increases. You can subscribe to project issue updates for any contributed module, theme, or translation.

Coding standards

Drupal Coding Standards

Note: The Drupal Coding Standards applies to code that is to become a part of Drupal. This document is based on the PEAR Coding standards.

Indenting

Use an indent of 2 spaces, with no tabs.

Control Structures

These include if, for, while, switch, etc. Here is an example if statement, since it is the most complicated of them:

```
if (condition1 || condition2) {  
    action1;  
}  
elseif (condition3 && condition4) {  
    action2;  
}  
else {  
    defaultaction;  
}
```

Control statements should have one space between the control keyword and opening parenthesis, to distinguish them from function calls.

You are strongly encouraged to always use curly braces even in situations where they are technically optional. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

For switch statements:

```
switch (condition) {  
    case 1:  
        action1;  
        break;  
    case 2:  
        action2;  
        break;  
    default:  
        defaultaction;  
        break;  
}
```

Function Calls

Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon. Here's an example:

```
$var = foo($bar, $baz, $quux);
```

As displayed above, there should be one space on either side of an equals sign used to assign the return value of a function to a variable. In the case of a block of related assignments, more space may be inserted to promote readability:

```
$short      = foo($bar);
$long_variable = foo($baz);
```

Function Declarations

```
function funstuff_system($field) {
  $system["description"] = t("This module insert funny text into posts randomly.");
  return $system[$field];
}
```

Arguments with default values go at the end of the argument list. Always attempt to return a meaningful value from a function if one is appropriate.

Comments

Inline documentation for classes should follow the Doxygen convention. More information about Doxygen can be found here:

- Document block syntax
- Comment commands

Note that Drupal uses the following docblock syntax:

```
/** 
 * Comments.
 */
```

And all Doxygen commands should be prefixed with a @ instead of a /.

Non-documentation comments are strongly encouraged. A general rule of thumb is that if you look at a section of code and think "Wow, I don't want to try and describe that", you need to comment it before you forget how it works.

C style comments /* */ and standard C++ comments // are both fine. Use of Perl/shell style comments (#) is discouraged.

Including Code

Anywhere you are unconditionally including a class file, use `require_once()`. Anywhere you are conditionally including a class file (for example, factory methods), use `include_once()`. Either of these will ensure that class files are included only once. They share the same file list, so you don't need to worry about mixing them - a file included with `require_once()` will not be included again by `include_once()`.

Note: `include_once()` and `require_once()` are statements, not functions. You don't need parentheses around the filename to be included.

PHP Code Tags

Always use `<?php ?>` to delimit PHP code, not the `<? ?>` shorthand. This is required for Drupal compliance and is also the most portable way to include PHP code on differing operating systems and setups.

Header Comment Blocks

All source code files in the core Drupal distribution should contain the following comment block as the header:

```
<?php
/* $Id$ */
```

This tag will be expanded by the CVS to contain useful information

```
<?php
/* $Id: CODING_STANDARDS.html,v 1.4 2004/10/27 11:55:32 uwe Exp $ */
```

Using CVS

Include the `Id` CVS keyword in each file. As each file is edited, add this tag if it's not yet present (or replace existing forms such as "Last Modified:", etc.).

The rest of this section assumes that you have basic knowledge about CVS tags and branches.

CVS tags are used to label which revisions of the files in your package belong to a given release. Below is a list of the required CVS tags:

DRUPAL-X-Y

(required) Used for tagging a release. If you don't use it, there's no way to go back and retrieve your package from the CVS server in the state it was in at the time of the release.

Example URLs

Use "example.com" for all example URLs, per RFC 2606.

Naming Conventions

Functions and Methods

Functions and methods should be named using lower caps and words should be separated with an underscore. Functions should in addition have the grouping/module name as a prefix, to avoid name collisions between modules.

Private class members (meaning class members that are intended to be used only from within the same class in which they are declared; PHP 4 does not support truly-enforceable private namespaces) are preceded by a single underscore. For example:

```
_node_get()  
$this->_status
```

Constants

Constants should always be all-uppercase, with underscores to separate words. Prefix constant names with the upcased name of the module they are a part of.

Global Variables

If you need to define global variables, their name should start with a single underscore followed by the module/theme name and another underscore.

Filenames

All documentation files should have the filename extension ".txt" to make viewing them on Windows systems easier. Also, the filenames for such files should be all-caps (e.g. README.txt instead of readme.txt) while the extension itself is all-lowercase (i.e. txt instead of TXT).

Examples: README.txt, INSTALL.txt, TODO.txt, CHANGELOG.txt etc.

Doxygen formatting conventions

Doxygen is a documentation generation system. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.

There is an excellent manual at the Doxygen site. The following notes pertain to the Drupal implementation of Doxygen.

General documentation syntax

To document a block of code, the syntax we use is:

```
/***
 * Documentation here
 **/
```

Doxygen will parse any comments located in such a block. Our style is to use as few Doxygen-specific commands as possible, so as to keep the source legible. Any mentions of functions or file names within the documentation will automatically link to the referenced code, so typically no markup need be introduced to produce links.

Documenting files

It is good practice to provide a comment describing what a file does at the start of it. For example:

```
<?php
/* $Id: theme.inc,v 1.202 2004/07/08 16:08:21 dries Exp $ */
/***
 * @file
 * The theme system, which controls the output of Drupal.
 *
 * The theme system allows for nearly all output of the Drupal system to be
 * customized by user themes.
 **/
```

The line immediately following the `@file` directive is a short description that will be shown in the list of all files in the generated documentation. Further description may follow after a blank line.

Documenting functions

All functions that may be called by other files should be documented; private functions optionally may be documented as well. A function documentation block should immediately precede the declaration of the function itself, like so:

```
/***
 * Verify the syntax of the given e-mail address.
 *
 * Empty e-mail addresses are allowed. See RFC 2822 for details.
 *
 * @param $mail
 *   A string containing an email address.
 * @return
 *   TRUE if the address is in a valid format.
 */
function valid_email_address($mail) {
```

The first line of the block should contain a brief description of what the function does. A longer description with usage notes may follow after a blank line. Each parameter should be listed with a @param directive, with a description indented on the following line. After all the parameters, a @return directive should be used to document the return value if there is one. Functions that are easily described in one line may omit these directives, as follows:

```
/**  
 * Convert an associative array to an anonymous object.  
 */  
function array2object($array) {
```

The parameters and return value must be described within this one-line description in this case.

Documenting hook implementations

Many modules consist largely of hook implementations. If the implementation is rather standard and does not require more explanation than the hook reference provides, a shorthand documentation form may be used:

```
/**  
 * Implementation of hook_help().  
 */  
function blog_help($section) {
```

This generates a link to the hook reference, reminds the developer that this is a hook implementation, and avoids having to document parameters and return values that are the same for every implementation of the hook.

Documenting themeable functions

In order to provide a quick reference for theme developers, we tag all themeable functions so that Doxygen can group them on one page. To do this, add a grouping instruction to the documentation of all such functions:

```
/**  
 * Format a query pager.  
 *  
 * ...  
 * @ingroup themeable  
 */  
function theme_pager($tags = array(), $limit = 10, $element = 0, $attributes = array()) {  
    ...  
}
```

The same pattern can be used for other functions scattered across multiple files that need to be grouped on a single page.

Comments

Inline documentation for classes should follow the PHPDoc convention, similar to Javadoc. More information about PHPDoc can be found here: <http://phpdoc.sourceforge.net/spec/howto.html>

Non-documentation comments are strongly encouraged. A general rule of thumb is that if you look at a section of code and think "Wow, I don't want to try and describe that", you need to comment it before you forget how it works.

C style comments (`/* */`) and standard C++ comments (`//`) are both fine. Use of Perl/shell style comments (`#`) is discouraged.

Indenting

Use an indent of 2 spaces, with no tabs. No trailing spaces.

PHP Code tags

Always use `<?php ?>` to delimit PHP code, not the `<? ?>` shorthand. This is required for Drupal compliance and is also the most portable way to include PHP code on differing operating systems and setups.

SQL naming conventions

- Don't use (ANSI) SQL / MySQL / PostgreSQL / MS SQL Server / ... Reserved Words for column and/or table names. Even if this may work with your (MySQL) installation, it may not with others or with other databases. Some references:
 - (ANSI) SQL Reserved Words
 - MySQL Reserved Words: 4.x, 3.23.x, 3.21.x
 - PostgreSQL Reserved Words
 - MS SQL Server Reserved Words

Some commonly misused keywords: `TIMESTAMP`, `TYPE`, `TYPES`, `MODULE`, `DATA`, `DATE`, `TIME`, ...

See also [bug] SQL Reserved Words.

- Capitalization, Indentation
 - UPPERCASE reserved words
 - lowercase (or Capitalize) table names
 - lowercase column names

Example:

```

SELECT r.rid, p.perm
FROM {role} r
  LEFT JOIN {permission} p ON r.rid = p.rid    -- may be on one line with prev.
ORDER BY name

```

- Naming

- Use plural or collective nouns for table names since they are sets and not scalar values.
- Name every constraint (primary, foreign, unique keys) yourself. Otherwise you'll see funny-looking system-generated names in error messages. This happened with the moderation_roles table which initially defined a key without explicit name as KEY (mid). This got mysqldump'ed as KEY mid (mid) which resulted in a syntax error as mid() is a mysql function (see [bug] mysql --ansi cannot import install database).
- Index names should begin with the name of the table they depend on, eg. INDEX users_sid_idx.

References:

- Joe Celko - Ten Things I Hate About You
- Joe Celko - SQL for Smarties: Advanced SQL Programming
- RDBMS Naming conventions
- SQL Naming Conventions

Functions

Functions should be named using lower caps and words should be separated with an underscore. Functions should have the grouping/module name as a prefix, to avoid name collisions between modules.

Constants

Constants should always be all-uppercase, with underscores to separate words. Prefix constant names with the upercased name of the module they are a part of.

Control structures

These include if, for, while, switch, etc. Here is an example if statement, since it is the most complicated of them:

```

if ((condition1) || (condition2)) {
  action1;
}
elseif ((condition3) && (condition4)) {
  action2;
}
else {
  defaultaction;
}

```

Control statements should have one space between the control keyword and opening parenthesis, to distinguish them from function calls.

You are strongly encouraged to always use curly braces even in situations where they are technically optional. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

For switch statements:

```
switch (condition) {
  case 1:
    action1;
    break;
  case 2:
    action2;
    break;
  default:
    defaultaction;
    break;
}
```

Header comment blocks

All Drupal source code files should start with a header containing the RCS \$Id\$ keyword:

```
<?php
// $Id: CODING_STANDARDS,v 1.1 2001/11/05 07:32:17 natrak Exp $
```

Note that everything after the starting \$Id and before the closing \$ is automatically generated by CVS - *you shouldn't edit this manually*. If you add a new file to CVS, just write // \$Id\$.

CVS

CVS (the *concurrent version system*) is a tool to manage software revisions and release control in a multi-developer, multi-directory, multi-group environment. It comes in very handy to maintain local modifications.

Thus, CVS helps you if you are part of a group of people working on the same project. In large software development projects, it's usually necessary for more than one software developer to be modifying modules of the code at the same time. Without CVS, it is all too easy to overwrite each others' changes unless you are extremely careful.

In addition, CVS helps to keep track of all changes. Therefore, the CVS server has been setup to mail all CVS commits to all maintainers. Thus, it does not require any effort to inform the other people about the work you have done, and by reading the mails everyone is kept up to date.

CVS concepts

Drupal uses Concurrent Versions System (CVS) to co-ordinate development, which can be thought of as a system for controlling the contents of a library. Describing CVS in terms of a library, books, and editions is only a metaphor - unlike a real, physical library, no matter how many people check out a book, it will always be available to the next person.

As with any library though, there are rules of behaviour towards other users and how you treat the library materials, no need to worry about overdue library tickets or leaving coffee stains on pages, but please be sure to act responsibly when contributing content to the library.

Repository

A repository can be thought of as a book, Drupal has two repositories which can be checked out (downloaded to your local computer):

- **Drupal**
the core Drupal code, i.e. what is downloaded as 'Drupal'.
- **Contributions**
modules, themes, translations, etc. supplied by contributors, i.e. all Drupal material that is not in the core.

Branch

CVS tracks different versions of content. Imagine different editions of a text book, each edition including amendments and additions - each edition is known as a 'branch', and has a branch tag to identify it, e.g. *Drupal 4.5*. Every branch corresponds to a particular version of Drupal that gets released.

Head

This is a special edition of the book (repository) which has not been published yet, or in CVS speak has not been "branched" yet. Think of "head" as a manuscript of the next edition of the book. Amendments and additions are added to the manuscript, it's proof-read and tested, and once it's ready it gets published (branched) as a new edition.

Working copy/Work area

Users can check out a copy of a book to work on locally, the original remains in the library and can be checked out by other users. When checking out a book remember that it is an edition (branch or head) of the book (repository). The copy of the book which is on the user's local computer is known as the "Working copy", "Work area" or "Work directory".

Changes the user makes to the local copy can be sent back to the library for inclusion in the repository.

Project

Modules, themes or translations can be added (committed) by users to the Contributions book (repository) at any time. It's possible to add to the manuscript (head) of Contributions, or any of the previous editions (branches). The website drupal.org tracks user additions to the Contributions book (repository) as "Projects". Each project has a description page from which it can be downloaded, and where issues and feature requests can be added by users.

Patch

Only a few Drupal developers are able to make changes directly to the Drupal book (repository). All other users who want to include changes they've made while working on their local copy of the Drupal book (repository), must create a file showing the differences between the local version and the version at drupal.org. This differences file is known as a "patch", and is sent to the rest of the development community by creating an "issue" at drupal.org and attaching the file.

Using CVS with branches and tags

To manage the different Drupal versions, we use tags and branches. A branch specifies a major Drupal version. For example, all 4.4.x versions belong in the DRUPAL-4-4 branch. Whenever we release a specific version, we create a tag. A tag is a marker which defines a snapshot of all the files in the CVS at a certain moment. For example, the tag DRUPAL-4-4-0 specifies all files at the time of the 4.4.0 release. The HEAD branch is special and is used to refer to the latest development version.

For an up-to-date complete list of branches and tags, see "Show files using tag:" at ViewCVS (at the bottom).

Here's a quick guide on using tags and branches. This assumes you have successfully checked out the 'main' and 'contributions' repositories. In my case, I usually make a folder in my home directory for each cvs server. In Drupal's case, cvs.drupal.org. For instance, you've made the CVS folder and here you check out your copy of the CVS version of Drupal.

```
~cvs.drupal.org$ cvs -d :pserver:anonymous@cvs.drupal.org:/cvs/drupal login  
~cvs.drupal.org$ cvs co drupal
```

This should leave you with a folder cvs.drupal.org/drupal which contains the current CVS code. You can keep this up-to-date by going into the cvs.drupal.org/drupal directory and using the command

```
~/cvs.drupal.org/drupal$ cvs update -dP
```

which will give you the latest copy, create any new directories that exist in the repository (-d), and trim unused directories (-P). Note that you don't need to specify the server at this point since the drupal directory contains a CVS folder that contains the repository and root information.

You've also done this for the contributions,

```
~cvs.drupal.org$ cvs -d :pserver:anonymous@cvs.drupal.org:/cvs/drupal-contrib login  
~cvs.drupal.org$ cvs co contributions
```

Which leaves you with the latest contributions in the `cvs.drupal.org/contributions` directory.

But now you want to have a nice copy of the 4.3.0 version, and you don't want to have to download the tgz file all the time. The CVS maintainer has branched the drupal repository and tagged it to keep track of this release. If you download it directly with the release tag it's going to overwrite your drupal folder. You probably want to keep it simple and use this command:

```
~cvs.drupal.org$ cvs -d :pserver:anonymous@cvs.drupal.org:/cvs/drupal -q checkout -d drupal-4.3 -r DRUPAL-4-3 drupal
```

That's going to create a new directory `cvs.drupal.org/drupal-4.3.0` that contains the 4.3.0 version. Once it's been checked out, you don't need to worry about specifying it again. The CVS directory in the `drupal-4.3.0` directory has the tag information along with repository and root information like we saw before. Just go into the `drupal-4.3` folder and execute

```
~/cvs.drupal.org/drupal-4.3$ cvs update -dP
```

to keep your copy of the 4.3.0 branch up to date.

Windows

You may have noticed this was geared towards the linux user. Sorry, I haven't used the windows clients for CVS. I'm sure they would work, I just haven't tried them (for very long). You could probably do this same thing on your windows box by installing one of the windows GUIs or using Cygwin, a GNU/UNIX client for windows. Probably the easiest way is to use TortoiseCVS.

Available Branches

The available branches currently are:

- HEAD
- DRUPAL-4-6
- DRUPAL-4-5
- DRUPAL-4-4
- DRUPAL-4-3

- DRUPAL-4-2
- DRUPAL-4-1
- DRUPAL-4-0
- DRUPAL-3-0

The DRUPAL-4-3-0 tag we used above is a marker in the DRUPAL-4-3 branch. Other tags for DRUPAL-4-3 are DRUPAL-4-3-1 and DRUPAL-4-3-2.

Apply for contributions CVS access

CVS GUIs and clients

There are a number of CVS 'front-ends' or GUIs which aim to improve on the command-line tools of CVS. These tools are grouped here by operating system/platform.

Cross-platform CVS clients

There are a number of GUI clients which run on multiple operating systems and platforms

Eclipse CVS plug-in

Perhaps the best CVS front-end I've used is the cross-platform tool Eclipse (www.eclipse.org). It has the functionality to do almost anything you can do with the command line, but you can use the console if you need to. As well, you can use it to edit the PHP code with another plugin.

CVS front ends for Windows

There are many CVS GUI front ends for Windows. Please add a new section if you know of additional ones

TortoiseCVS

TortoiseCVS lets you work with files under CVS version control directly from Windows Explorer. It's freely available under the GPL. The following tutorial teaches how to use TortoiseCVS with Drupal.

- Download TortoiseCVS from <http://www.tortoisecvs.org/download.shtml> and install it.
- In Windows Explorer, select the folder under which you want the Drupal source directory to live. Right-click on it. There are two new sections in the context menu - **CVS Checkout** and **CVS >**. Select **CVS Checkout**.
- Fill in the following fields:

Protocol: Password server (:pserver:)
Server: [cvs.]drupal.org
Repository folder: /cvs/drupal (main distro) or /cvs/drupal-contrib
(contributions)
User name: anonymous
Module: drupal (main distro) or
contributions
(contributions)

and press "OK".

- You will be asked for password. Enter anonymous and press "OK".
- A log window which monitors the checkout process will appear. Checking out the whole CVS repository will take a while.
- If everything works, you will see the message "Success, CVS operation completed" at the end of the log. A new directory (named like the module selected before) with the sources will be created.
- To bring your Drupal source tree up-to-date, select it's root folder ("drupal" / "contributions"), right-click it and do a "CVS Update".

The process above retrieves the freshest files from the repository (the so-called HEAD branch). These are sometimes unstable. To get Drupal modules and themes that are stable and ready for production (which you can also download from the Drupal downloads page), follow the process described above, but before hitting "OK" you need to:

- Click on the "Revision" tab on the CVS checkout dialog.
- Enable "Get tag/branch".
- Enter DRUPAL-4-1-0 or DRUPAL-4-00 depending on the version you are using in the tag/branch field.
- Hit OK.

You can also generate patch files with TortoiseCVS. Just select the files which you have patched in Windows Explorer. Then right click into the CVS => Make Patch menu item. Then you may wish to read Creating and sending your patches

WinCVS

WinCVS is another graphical CVS client available for MS Windows and for Macs. You can download the latest version from <http://www.wincvs.org/>. The checkout / update process is similar to the one described above.

CVS on Mac OS X

There are a number of good CVS clients for Mac OS X.

CVL: point and click CVS

For Mac users who are unused to the command line, CVS can at first look a bit daunting. Fortunately there is an application called CVL that provides control of CVS through a point and click interface. Most of the instructions for using CVL will also apply when using other applications to control CVS.

Setting up/step by step CVS

Here is a step-by-step guide to installing and setting up CVL.

1. **Install CVS.** The first step of using any application is of course to install it - CVS is installed by default with the Apple Developer tools, so if you haven't installed these yet, download the latest version and install them. They also include a lot of other useful stuff like Project Builder and File Merge (Apple Developer Membership is required, but free).

2. Install CVL

The CVL package and complete installation instructions are available at <http://www.sente.ch/software/cvl/>

3. Setup CVL to work with CVS.

Set CVS to ignore the 'hidden' .DS_Store files which OS X creates in each folder. To do this you need to open the Terminal
(Application->Utilities->Terminal), and type the following:

cd

takes you to root of your account

pico

opens the Pico text application

.DS_Store

specifies which file types you want CVS to ignore

Now press the keys: **Control** and **x** at the same time

This closes the document you've just written, it will ask you if you want to save it - press **y** for yes, then type in the name of the file **.cvignore** (note the '.') and press return. You've finished with the Terminal, so you can quit it.

4. Create a folder to put the CVS files into. The best place to do this is in the 'Sites' folder, to make it easy to use them through the Apache server built into your system. You can name the folder anything you want, my one is called 'drupal_cvs'.
5. **Step five** open the CVL application, you now need to Checkout (download) the latest version of the Drupal CVS like this:

Tools->Repositories->Show Repositories

Click Add

Choose a repository dialog box will appear.

In **Repository type** choose **pserver**.

CVS User: your **Username** (that you applied for CVS with)

Host: **cvs.drupal.org**

Path: **/cvs/drupal** (main distro) or **/cvs/drupal-contrib** (contributions)

Password: your **password** (that you applied for CVS with)

Click **Add**.

Next go to Tools->Repositories->Show Repositories

The Drupal repository is now listed in the **Repositories** window. Select it and press **Checkout...**

Checkout Module dialog box appears.

Choose Module: **drupal** (main distro) or **contributions** (contributions)

New work area location: **Choose...** select the folder you created in **step four**.

Press **Checkout**.

Wait patiently, this may take some time, as the whole of the Repository needs to be downloaded - you can see this happening if you open the console window (Tools->Console->Show Console), don't worry if you don't see anything at first, CVL usually thinks about what it's doing for a minute or two before taking action.

When this is finished you will have a copy of the Drupal repository files in the folder you created on your hard drive, this is your **Work Area**, where you work on projects before uploading them to the repository for others to use.

Basic CVS with CVL

You now have a **Work Area** on your hard drive which is a mirror of the Repository on the Drupal server. You can see this by using the CVL menu Work Area->Open Recent and selecting the repository you just downloaded (drupal or contributions).

You can use this work area in the same way you would any other folder on your hard drive - create new files with BBEdit (or whatever you use), drag files to the trash, add new folders, delete folders - it's just a regular folder.

Once you've done some work you want to upload back to the Drupal server here's what you do:

Update the CVS by selecting the folder the new work is in, then **Control+Click** on the folder and choose **Update** from the contextual menu that pops up (or through the menu File->Update). CVS now shows any new files or folders that you have added (with a blue * in front).

Next you need to tell CVS to mark the files and folders for upload next time you send your changes to the Drupal repository. To do this select the files and folders and **Control+Click**, choose **Add To Work Area** (or through the menu File->Add To Work Area).

To upload your work to the Drupal repository, select your files and folders and **Control+Click**, choose **Commit...** (or through the menu File->Commit...). CVS will now add your work to the Drupal repository.

Preparing a project

New module, theme or translation projects should be started in the Drupal CVS contributions repository.

In the Finder, go to the folder where you saved the CVS contributions working copy, and create a new folder in the appropriate subfolder. For example, if you are working on a new module, create new folder in the module folder. Name the new folder to whatever you want to call the project. Try to make the name short and descriptive. Avoid spaces, use "_" to separate words, but read the Developer guidelines to understand how underscores in module names may interact with code behaviour.

In CVL, open the CVS contributions work area, navigate to the folder containing the new folder you just created, control-click on it and select "Refresh" from the menu that pops up.

The new folder, and any files you put in it, should now show up in CVL with a blue * next to it.

The blue * signifies that the files have not been added to the work area yet.

In CVL select the new folder, control-click on it and select "**Mark File(s) for Addition**" from the menu that pops up.

The blue * will now change to a green + next to each of the new files and folders, signifying that the files are part of the working copy and can be added to the repository at drupal.org once you want to commit them.

Committing a project

Once your new module, theme or translation is complete you may want to add it to Drupal's contributed repository, and create a project for it at www.drupal.org.

1. Add your project to CVS

Your new project should first be added to the trunk of the contrib repository, which is known as the 'cvs' or 'HEAD' branch. (see Setting up)

Add the files to the 'cvs' version of the contrib repository (see Preparing a project). Once added to CVS, the project folder and each of its files will have a green '+' next to it, this means they are ready to be committed.

Select the project's folder, for example:

- **banana.module**
contrib/modules/banana/banana.module
select folder 'banana'

With the project folder selected, control-click, select 'Commit...'

A dialog box will appear into which you can type a log message. The log message should briefly explain what new features have been added to this version of the files or what bugs have been fixed.

2. Add your project to the Drupal Project tracker

Your files are now in the contrib repository, now you need to make drupal.org aware of your new project.

By creating a 'project' at www.drupal.org the files in CVS become available for download on the 'Downloads' page, it also allows users to submit feature requests and bug reports for the project.

Log in to www.drupal.org, in the side account block click on "create content", then click "project".

Fill in the project form page to create the new project. The project will appear on drupal.org in a day or two.

Drupal CVS repositories

Main repository

There are two ways to access the latest Drupal sources in the main CVS repository. If you just want to have a quick look at some files, use the ViewCVS web interface. If you need the complete source tree to study and work with the code, follow these steps:

- If you don't have it yet, install a recent copy of CVS. If you are on Windows, you may check CVS front ends for Windows). Mac OS X users may find the tutorial on the CVL front end for CVS in section CVL: point and click CVS helpful.
- Login by running the command:

```
$ cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal login
```

The required password is 'anonymous' (without the quotes).

- To check out the latest drupal sources, run the command:

```
$ cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal checkout drupal
```

This will create a directory called `drupal` containing the latest drupal source tree.

- Once you have a copy of the Drupal source tree, use

```
$ cvs update -dP
```

in the source root dir to update all files to it's latest versions (-d: Create any (new) directories that exist in the repository if they're missing from the working directory. -P: Prune empty directories - directories that got removed in the repository will be removed in your working copy, too).

If you can't or don't want to use CVS, you can download nightly CVS snapshots from <http://drupal.org/files/projects/drupal-cvs.tar.gz>.

Contributions repository

The Contributions repository is a separate CVS repository where people can submit their modules, themes, translations, etc. See the contributions FAQ.txt and README.txt for more information.

As the Main repository, you can browse it via the web interface. For anonymous (read-only) access, do the following:

- Login by running the command

```
$ cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal-contrib login
```

The required password is 'anonymous' (without the quotes).

- To check out the latest drupal contributions, run the command:

```
$ cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal-contrib checkout contributions
```

- To check out contributions for a certain Drupal version, do

```
$ cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal-contrib checkout  
-r <version tag> contributions
```

where *<version tag>* is one of the tags listed under "Q: How do I control the releases of my module/theme?" here.

- To update your tree to the latest version, do

```
$ cvs update -dP
```

in the source root dir.

If you want to add your own modules, themes, translations, etc., you need CVS write access:

Promoting a project to be an official release

To promote a project from the HEAD of the CVS tree to an official release state, the author needs to move the CVS tag.

For instance, to promote the CVS HEAD of the French translation to the official 4.5 release, using the command line from within inside the contributions/translation/fr, just do:

```
$ cvs up -dA  
$ cvs tag -F DRUPAL-4-5
```

Adding a file to the CVS repository

If you would like to add a file or a directory, first you need to download the parent directory. Once again: the *parent* directory, not the directory you want to add something to, but the parent of it. I can not find any logic in this, but this is so.

First, issue the following command:

```
export  
CVSROOT=:pserver:icvslogin:cvspasswd@cvs.drupal.org:/cvs/drupal-contrib
```

To add a new file to a module:

```
cvs co contributions/modules/modulename
cd contributions/modules
cp sourcefile modulename
cvs add modules/sourcefilename
cvs commit
```

Say you want to create a sandbox named mysandbox. Then do the following:

```
cvs co contributions/sandbox/weblinks [1]
cd contributions
mkdir sandbox/mysandbox
cvs add sandbox/mysandbox
cvs commit
```

[1] does not really matter which directory, just sg. from sandbox. I like this one, because it is small.

Of course, you may do several things in one commit, adding files, removing files, updating files. Neither remove (cvs remove) nor update (cvs update) is such a tedious process. Warning: you need to check out with your CVS account, because CVS ignores CVSROOT for existing checkouts.

Tracking Drupal source with CVS

Note: The following assumes you have both basic knowledge of CVS and your own local repository set up and working.

If you've been modifying the Drupal source code for your own purposes (or developing a module or theme) and manually applying your changes to the Drupal source every time it updates, you may be glad to learn that CVS can help make this easier.

This is usually referred to as 'tracking third-party sources' and requires knowledge of the CVS concepts branching, release tags, and the vendor tag. We'll work through an example here and explain these concepts as we go.

Example

Lets assume we'd like to track current Drupal CVS HEAD, and start by downloading the source. In this case we'll export using anonymous CVS (we could also just download a tarball).

Begin by logging in to the anonymous CVS server, the required password is 'anonymous':

```
cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal login
```

Then export the newest development version of drupal using the HEAD release tag:

```
cvs -d:pserver:anonymous@cvs.drupal.org:/cvs/drupal export -r HEAD drupal
```

Now that we have a local copy of the drupal source we can import it into our own CVS repository. In this example we import with a log message including the date '-m "message text"', a module location/name of 'sites/drupal' (customize that to suit your own CVS repository), a vendor tag of 'drupal' and a release tag of 'HEAD20040110'. We also use the -ko option to prevent keyword expansion (this preserves the CVS \$Id\$ tags used on drupal.org):

```
cd drupalcvs import -ko -m "Import CVS HEAD on Jan 10th 2004" sites/drupal drupal HEAD20040110
```

Before we can customize we need to checkout into a working directory. Then we can modify a file or files and commit:

```
cvs checkout drupal cd drupal...modify a file or files...cvs commit
```

We now have a drupal module with a special 'vendor branch' (identified by the vendor tag), which contains the drupal source files we imported, and a main trunk with our modified files. Any files modified at this point are now HEAD on the main trunk of the module, whilst the unmodified files remain HEAD on the vendor branch (HEAD being what is produced by cvs update). For an individual file (fileone.php) the version history now looks like something like this:

```
          HEAD
          +---+
[Main trunk]  fileone.php *-----+ 1.2 +
              \           +---+
              +---+
[Vendor Branch]  + 1.1.1.1 +
              +---+
              (tag:HEAD20040110)
```

Updating the vendor branch

At some later point the drupal source code will have been updated and we'll want to add the updated version to our repository. We do this by repeating the process described above, we get a fresh copy of the source from drupal.org, and import using the same vendor tag but change the release tag from 'HEAD20040110' to reflect the newer version:

```
cvs import -ko -m "Import CVS HEAD on Jan 11th 2004" sites/drupal drupal HEAD20040111
```

This updates the vendor branch, a single files revision history can now appear four different ways, depending on whether it has been modified by us, by the vendor (drupal.org), by both, or not at all.

If the file was modified only by us, our modified version remains the head revision:

```

          HEAD
          +---+
[Main trunk] fileone.php *-----+ 1.2 +
          \      +---+
          \      +---+
          +-----+
[Vendor Branch] + 1.1.1.1 +
          +-----+
          (tag:HEAD20040110)

```

If the file was modified only by the vendor, the new version becomes the HEAD revision:

```

[Main trunk] filetwo.php *
          \
          \
          HEAD
          +-----+
[Vendor Branch] + 1.1.1.1 +-----+ 1.1.1.2 +
          +-----+           +-----+
          (tag:HEAD20040110)   (tag:HEAD20040111)

```

And if the file was modified by both us and the vendor:

```

          HEAD
          +---+
[Main trunk] filethree.php *-----+ 1.2 +
          \      +---+
          \      +---+
          +-----+           +-----+
[Vendor Branch] + 1.1.1.1 +-----+ 1.1.1.2 +
          +-----+           +-----+
          (tag:HEAD20040110)   (tag:HEAD20040111)

```

Our version of filethree.php remains the HEAD revision, but this is clearly not desirable since it doesn't carry the latest changes. In fact, during our import of the latest source CVS would have warned us of conflicts between the two versions of filethree.php, we need to merge the changes to remove this conflict:

```
cvs checkout -jHEAD20040110 -jHEAD20040111 drupal
```

Examine the merged file to ensure the changes CVS made were sane and then 'cvs commit' the changes back to the main trunk. Leaving us with a new revision which becomes HEAD:

```

          HEAD
          +---+           +---+
[Main trunk] filethree.php *-----+ 1.2 +-----+ 1.3 +
          \      +---+           +---+
          \      +---+
          +-----+           +-----+
[Vendor Branch] + 1.1.1.1 +-----+ 1.1.1.2 +
          +-----+           +-----+
          (tag:HEAD20040110)   (tag:HEAD20040111)

```

Summary

It should now be clear that using the CVS vendor tag to create a vendor branch in your own drupal module you can track changes to the drupal source code whilst also maintaining and developing your own customizations and new features for drupal. This example has been kept very simple for the purposes of explanation, but the basic process can be used to achieve many different things, some examples:

- Track a specific release of Drupal (e.g. 4.3, or 4.2), instead of the development (CVS HEAD) version.
- Maintain your customized sites with modules, themes, static pages, images etc all added to your CVS repository, whilst still tracking and importing updates to the drupal core.
- Branch your module to maintain several customized web sites off a single tracked branch of the drupal core.

Reading the following additional resources is highly recommended.

Additional resources

- Article by Nick Patavalis: The mechanics and a methodology for tracking 3rd party sources with CVS
- The section “Tracking third-party sources” in the CVS manual
- The section “Tracking third-party sources” in a book on CVS

Sandbox maintenance rules

1. Always document your changes.
2. Split different set of patches into different directories. It takes longer to find the set of files relating to one change if it is mixed in with 2 other patches.
3. Keep the documentation current. Try to keep some track of your reasoning too. If I read in a README that change X wasn't a good idea after all it makes the reviewer wonder why.
4. Document the status of your patch. It is important to know if this is an early test, or considered stable and workable but the author of the patch.
5. All patches should be against the latest CVS version of Drupal, and include in the README when it was last synced.
6. Don't use a sandbox for developing modules. There is a different directory structure for that.
7. If your patch is 4 lines long don't bother to put it in a sandbox. Just mail it to the devel list and find out quicker if people like it or not. Small patches are quick to check and find out if work. Sandboxes should be for more extensive changes.
8. Try to maintain patches in the sandbox. They are so much easier to check than compete files. If you are using CVS then you can use diff (cvs -H diff)
9. Please make sure your script passes the code-style.pl script. It isn't perfect, and sometimes a bit too strict, but it will ensure some level of compliance with

the coding standards.

Additional references

- CVS book
- CVS docs
- CVS FAQ
- CVS guide from TLDP

Drupal's APIs

If you are interested in developing Drupal modules or hacking away at the Drupal core then this is the place to find details about all the functions and classes defined in Drupal.

We now use Doxygen to automatically generate documentation from the latest drupal sources. This allows us to ensure that documentation is up-to-date, and to simultaneously track multiple versions of the documentation.

API Documentation is available from drupaldocs.org for:

- Drupal CVS HEAD
- Drupal 4.5.x
- Drupal 4.4.x

Please also read the Drupal Coding Standards page, which contains some guidelines for writing Doxygen comments.

Drupal.org site maintainers

Below is an alphabetical list of users who have additional permissions to help maintain the drupal.org website:

1. adrian
2. ahoppin
3. aldon@deanspace.org
4. Amazon
5. andremolnar
6. ax
7. BÃ¶r Kessels
8. bertboerland@ww...
9. blogdiva@www.cu...
10. Boris Mann
11. bryan kennedy
12. cel4145
13. chx

14. Development Seed
15. dgreenberg
16. Dries
17. drumm
18. Dublin Drupaller
19. ericgundersen
20. FactoryJoe@civi...
21. Gerhard Killesreiter
22. Goba
23. JonBob
24. Junyor
25. jvandyk
26. kbahey
27. Kieran Huggins
28. kika
29. killies@www.drop.org
30. Kjartan
31. Kobus
32. mathias
33. Morbus Iff
34. moshe weitzman
35. nysus
36. puregin
37. Richard Eriksson
38. rivena
39. Robert Castelo
40. robertDouglass
41. Robin Monks
42. Roland Tanglao@...
43. sepeck
44. Steven
45. TDobes
46. Uwe Hermann
47. walkah
48. wnorrix

If you have been around for a while, and you want to help maintain Drupal.org, get in touch with Dries.

Site maintainer's guide

This page lists some guidelines for Site Maintainers on Drupal.org.

Unpublishing vs deleting of content

You should only unpublish a post if you can conceivably imagine it being re-published in the future. This should only be for very rare cases. A good example is an unmaintained project (someone might take over development later): unpublish, don't delete (*). On the other hand, spam can be deleted immediately.

(*) Actually, old projects are automatically unpublished by the cvs scripts, so this is not something you need to do.

Blocking vs deleting of users

Deleting users is a very destructive action, as it makes all their content inaccessible in most places, even to administrators. It should not be done. If a user is a troublemaker, just block their account (click username -> edit -> status: blocked). Of course, you should not block people just because they say unfavorable things about Drupal. Here are good reasons to block someone:

- Spaming (even once)
- Repetitive flaming
- Repetitive posting of trash content (test posts, inappropriate book pages, ...)

Suggested Workflow

When you spot something out of the ordinary, we suggest these steps:

1. Take a look at the user's post history on the "user -> track -> track posts" page. This will possibly show more bad posts by the same person.
2. Take a look at the user's page visit history on the "user -> track -> track page visits" page. That way, you can easily tell if a user just registered to spam or if they made only one bad post in a series of good ones.
3. If the content is spam, you can delete it immediately and block the person's account. Otherwise, send them a note through their contact tab about it:

Your post Foobar on <http://drupal.org/node/1234> was inappropriate because it contained flaming. Please be nice to your fellow visitors on Drupal.org, or your account may be blocked.

4. If you know someone to be a troublemaker who has been warned before, block their account.

Badly formatted posts

If you see a post with bad formatting which messes up the page's layout, please edit it. A common mistake for newbies is to use two opening tags rather than an opening/closing pair. Tags like bold and italic can 'bleed through' beyond the post, while unclosed block-level tags can mess up the positioning of the sidebar.

If someone made a serious mistake while posting a forum topic and posted a correction in a comment below, try to update the original post and delete the correction.

Drupal test suite

Drupal is currently lacking some test suite to be run by developers before submitting important patches. The following setup isn't really a test suite but it is a start to avoid the most embarrassing errors. A more complete solution would be unit tests as proposed by Moshe Weitzman. but they'd also be a lot more work.

Ok, here is what I will do in the future:

1. Enable the menu module and disable the 'log out' link.
2. Run

```
wget --mirror --delete-after  
http://killes.drupaldevs.org/  
where killes.drupaldevs.org is my development site. You can add --wait=5 to  
the options if you don't want a free stress test.
```

3. If I want to test as an authenticated user I do

```
wget --mirror --delete-after  
--load-cookies=/path/to/cookies.txt  
http://killes.drupaldevs.org/  
where /path/to/cookies.txt is the cookie inside my .mozilla directory.
```

Note that this can take some time. wget will access every Drupal page linked from the frontpage. You can later have a look at the error logs and find out if any errors where caused.

FAQ

This FAQ (Frequently Asked Questions) collects questions of interest to Drupal Contributors.

PHP Debugger

What do you folks use for debugging PHP? I'm getting \$conf and header errors on my Drupal installation and would like to track them down, learning Drupal in the process. Is there a debugger for PHP where I can set breakpoints, see values change, etc.? Thanks for your help.

Module developer's guide

Developer documentation can be found at <http://drupaldocs.org/> and in the remainder of the Drupal developer's guide below.

- drupaldocs.org documents the Drupal APIs and presents an overview of Drupal's building blocks along with handy examples.
- The Drupal developer guide provides guidelines as how to upgrade your modules (API changes) along with development tips/tutorials.

Introduction to Drupal modules

When developing Drupal it became clear that we wanted to have a system which is as modular as possible. A modular design will provide flexibility, adaptability, and continuity which in turn allows people to customize the site to their needs and likings.

A Drupal module is simply a file containing a set of routines written in PHP. When used, the module code executes entirely within the context of the site. Hence it can use all the functions and access all variables and structures of the main engine. In fact, a module is not any different from a regular PHP file: it is more of a notion that automatically leads to good design principles and a good development model. Modularity better suits the open-source development model, because otherwise you can't easily have people working in parallel without risk of interference.

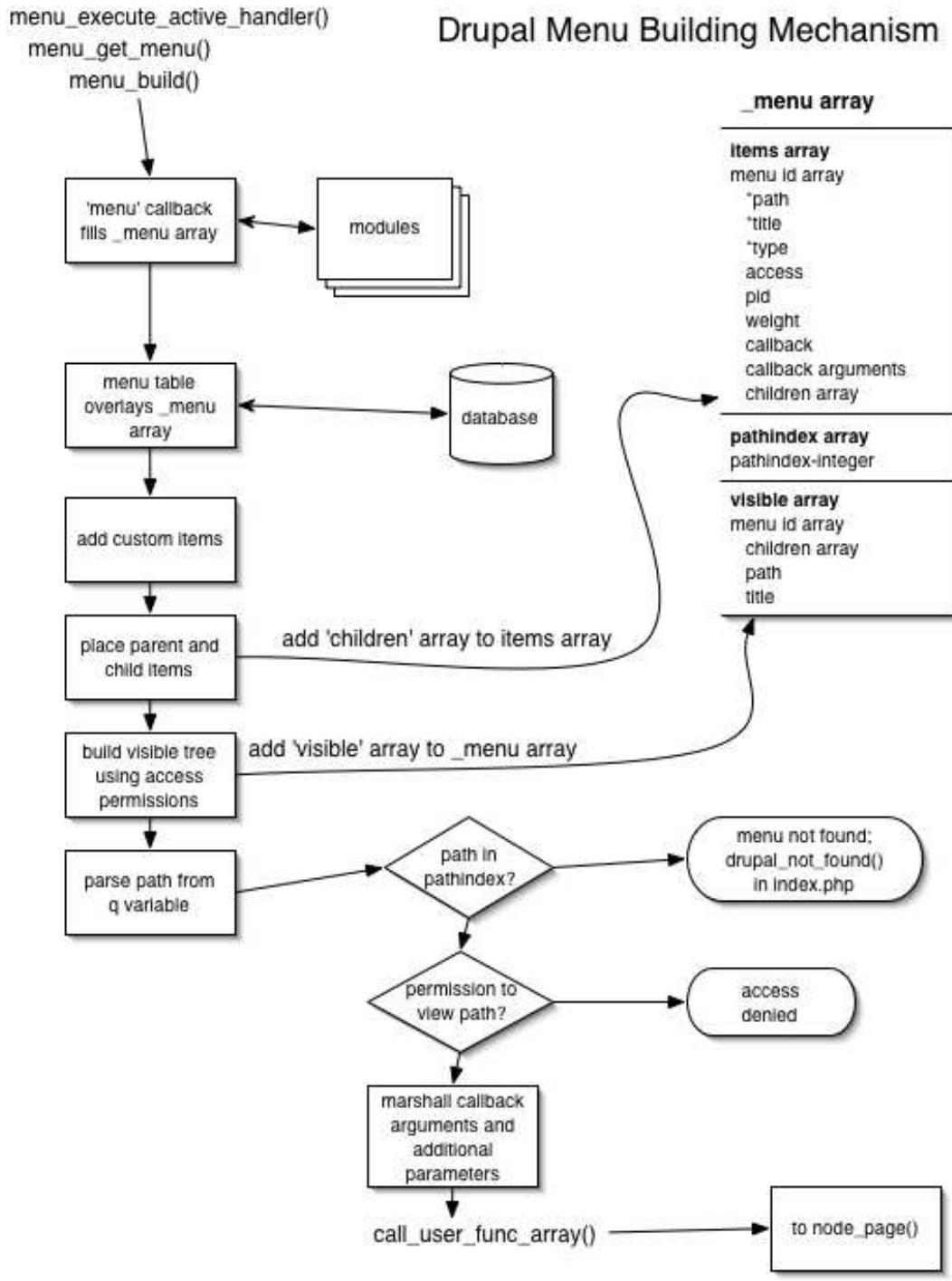
The idea is to be able to run random code at given places in the engine. This random code should then be able to do whatever needed to enhance the functionality. The places where code can be executed are called "hooks" and are defined by a fixed interface.

In places where hooks are made available, the engine calls each module's exported functions. This is done by iterating through the modules directory where all modules must reside. Say your module is named foo (i.e. modules/foo.module) and if there was a hook called bar, the engine will call foo_bar() if this was exported by your module.

See also the overview of module hooks, which is generated from the Drupal source code.

Drupal's menu building mechanism

(Note: this is an analysis of the menu building mechanism in pre-4.5 CVS as of August 2004. It does not include menu caching.)



by John VanDyk

This continues our examination of how Drupal serves pages. We are looking specifically at how the menu system works and is built, from a technical perspective. See the excellent overview in the menu system documentation.

We begin in index.php, where `menu_execute_active_handler()` has been called. Diving in from `menu_execute_active_handler()`, we immediately set the `$menu` variable by calling `menu_get_menu()`. The latter function declares the global `$_menu` array (note the underline, it means a 'super global', which is a predefined array in PHP lore) and calls `_menu_build()` to fill the array, then returns `$_menu`. Although `menu_get_menu()` initializes the `$_menu` array, the `_menu_build()` function actually reinitializes the `$_menu` array. Then it sets up two main arrays within `$_menu`: the `items` array and the `path` index array.

The `items` array is an array keyed to integers. Each entry contains the following fields:

| Required fields | | |
|------------------------|---------|---|
| path | string | the partial URL to the page for this menu item |
| title | string | the title that this menu item will have in the menu |
| type | integer | a constant denoting the menu item type (see comments in menu.inc) |
| Optional fields | | |
| access | boolean | |
| pid | integer | |
| weight | integer | |
| callback | string | name of the function to be called if this menu item is selected |
| callback arguments | array | |

An array called `$menu_item_list` is populated by sending a 'menu' callback to all modules with 'menu' hooks (that is, they have a function called `foo_menu()` where `foo` is the name of the module). So each module has a chance to register its own menu items. It is interesting that when the node module receives the menu callback through `node_menu()`, and the path is something like 'node/1' as it is in our present case, the complete node is actually loaded via the `node_load()` function so it can be examined for permissions. The `$node` variable into which it was loaded then goes out of scope, so the node is gone and needs to be rebuilt completely later on. This seems like a golden opportunity for the node module to cache the node.

The `$menu_item_list` array is normalized by making sure each array entry has a path, type and weight entry. As each entry is examined, the path index array of the `$_menu` array is checked to see if the path of this menu item exists. If an equivalent path is already there in the path index array, it is blasted away. The path index of this menu item is then added as a key with the value being the menu id. In the `items` array of the `$_menu` array, the menu id is used as the key and the entire array entry is the value.

Note: the \$temp_mid and \$mid variables seem to do the same thing. Why, syntactically, cannot only one be used?

The path index array contained 76 items when serving out a simple node with only the default modules enabled.

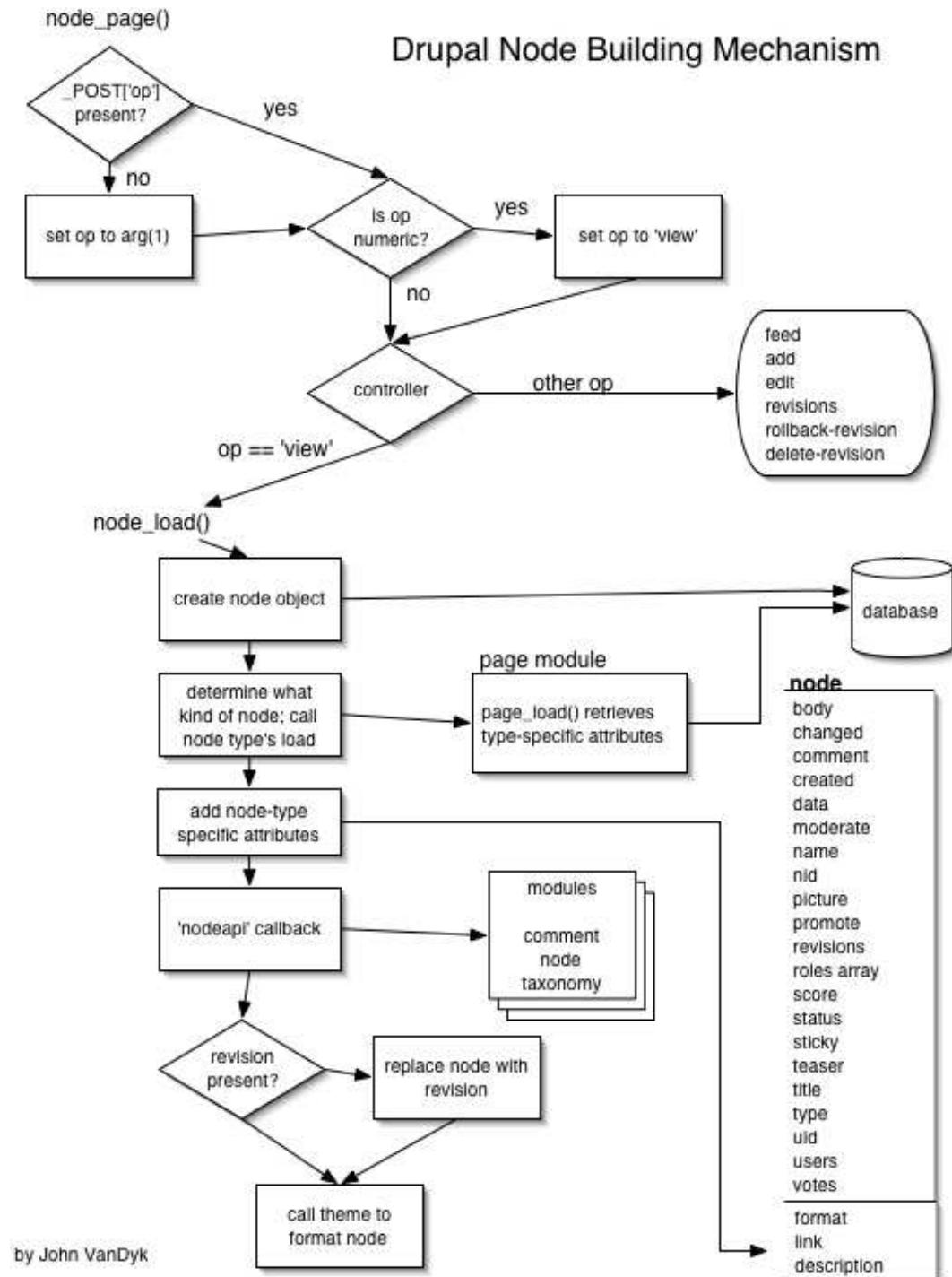
Next the menu table from the database is fetched and its contents are used to move the position of existing menu items from their current menu ids to the menu ids saved in the database. The comments says "reassigning menu IDs as needed." This is probably to detect if the user has customized the menu entries using the menu module. The path index array entries generated from the database can be recognized because their values are strings, whereas up til now the values in the path index array have been integers.

Now I get sort of lost. It looks like the code is looking at paths to determine which menu items are children of other menu items. Then _menu_build_visible_tree is a recursive function that builds a third subarray inside \$_menu, to go along with items and path index. It is called visible and takes into account the access attribute and whether or not the item is hidden in order to filter the items array. As an anonymous user, all items but the Navigation menu item are filtered out. See also the comments in menu.inc for menu_get_menu(). In fact, read all the comments in menu.inc!

Now the path is parsed out from the q parameter of the URL. Since node/1 is present in the path index, we successfully found a menu item. It points to menu item -44 in our case, to be precise, but there must be a bug in the Zend IDE because it shows item -44 as null. Anyway, the menu item entry is checked for callback arguments (there are none) and for additional parameters (also none), and execution is passed off to node_page() through the call_user_func_array function.

Drupal's node building mechanism

(This walkthrough done on pre-4.5 CVS code in August 2004.)



The node_page controller checks for a `$_POST['op']` entry and, failing that, sets `$op` to `arg(1)` which in this case is the '1' in `node/1`. A numeric `$op` is set to `arg(2)` if `arg(2)` exists, but in this case it doesn't ('1' is the end of the URL, remember?) so the `$op` is hardcoded to 'view'. Thus, we succeed in the 'view' case of the switch statement, and are shunted over to `node_load()`. The function `node_load()` takes two arguments, `$conditions` (an array with `nid` set

to desired node id -- other conditions can be defined to further restrict the upcoming database query) for which we use `arg(1)`, and `$revision`, for which we use `_GET['revision']`. The 'revision' key of the `_GET` array is unset so we need to make brief stop at `error_handler` because of an undefined index error. That doesn't stop us, though, and we continue pell-mell into `node_load` using the default `$revision` of -1 (that is, the current revision). The actual query that ends up being run is

```
SELECT n.*, u.uid, u.name, u.picture, u.data FROM node n INNER JOIN users u  
on u.uid WHERE n = '1'
```

We get back a joined row from the database as an object. The data field from the `users` table is serialized, so it must be unserialized. This data field contains the user's roles. How does this relate to the `user_roles` table? Note that the comment "`// Unserialize the revisions and user data fields`" should be moved up before the call to `drupal_unpack()`.

We now have a complete node that looks like the following:

| Attribute | Value |
|-----------|--|
| body | This is a test node body |
| changed | 1089859653 |
| comment | 2 |
| created | 1089857673 |
| data | a:1:{s:5... (serialized data)} |
| moderate | 0 |
| name | admin |
| nid | 1 |
| picture | " |
| promote | 1 |
| revisions | " |
| roles | array containing one key-value pair, 0 = '2' |
| score | 0 |
| status | 1 |
| sticky | 0 |
| teaser | This is a test node body |
| title | Test |
| type | page |
| uid | 1 |
| users | " |
| votes | 0 |

All of the above are strings except the roles array.

So now we have a node loaded from the database. It's time to notify the appropriate module that this has happened. We do this via the `node_invoke($node, 'load')` call. The module called via this callback may return an array of key-value pairs, which will be added to the node above.

The `node_invoke()` function asks `node_get_module_name()` to determine the name of the module that corresponds with the node's type. In this case, the node type is a page, so the `page.module` is the one we'll call, and the specific name of the function we'll call is `page_load()`.

If the name of the node type has a hyphen in it, the left part is used. E.g., if the node type is page-foo, the page module is used.

The page_load() function turns out to be really simple. It just retrieves the format, link and description columns from the page table. The 'format' column specifies whether we're dealing with a HTML or PHP page. The 'link' and 'description' fields are used to generate a link to the newly created page, however, those will be deprecated with the improved menu system. To that extend, the core themes no longer use this information (unlike some older themes in the contributions repository). We return to node_load(), where the format, link and description key-value pairs are added to the node's definition.

Now it's time to call the node_invoke_nodeapi() function to allow other modules to do their thing. We check each module for a function that begins with the module's name and ends with _nodeapi(). We hit paydirt with the comment module, which has a function called comment_nodeapi(&\$node, \$op, arg = 0). Note that the node is passed in by reference so that any changes made by the module will be reflected in the actual node object we built. The \$op argument is 'load', in this case. However, this doesn't match any of comment_nodeapi()'s symbols in its controller ('settings', 'fields', 'form admin', 'validate' and 'delete' match). So nothing happens.

Our second hit is node_nodeapi(&\$node, \$op, \$arg = 0) in the node.module itself. Again, no symbols are matched in the controller so we just return.

We'll try again with taxonomy_nodeapi(&\$node, \$op, \$arg = 0). Again, no symbols match; the taxonomy module is concerned only with inserts, updates and deletes, not loads.

Note that any of these modules could have done anything to the node if they had wished.

Next, the node is replaced with the appropriate revision of the node, if present as an attribute of \$node. It is odd that this occurs here, as all the work that may have been done by modules is summarily blown away if a revision other than the default revision is found.

Finally, back in node_page(), we're ready to get down to business and actually produce some output. This is done with the statement

```
print theme('page', node_show($node, arg(3)), $node->title);
```

And what that statement calls is complex enough to again warrant another commentary. (Not yet done.)

How Drupal handles access

I believe this page should explain how user_access table works.

- 1.- Drupal checks if the user has access to that module, if he does ...
- 2.- Then he checks the user_access page where gid is the role, view should be 1 and realm should be "all". If there is no access given in that table, he will not give the access to the user.

I believe there is not enough documentation on how to use node access, and hopefully this page will have more information as people contribute.

Drupal's page serving mechanism

This is a commentary on the process Drupal goes through when serving a page. For convenience, we will choose the following URL, which asks Drupal to display the first node for us. (A node is a thing, usually a web page.)

`http://127.0.0.1/~vandyk/drupal/?q=node/1`

A visual companion to this narration can be found here; you may want to print it out and follow along. Before we start, let's dissect the URL. I'm running on an OS X machine, so the site I'm serving lives at `/Users/vandyk/Sites/`. The drupal directory contains a checkout of the latest Drupal CVS tree. It looks like this:

```
CHANGELOG.txt
cron.php
CVS/
database/
favicon.ico
includes/
index.php
INSTALL.txt
LICENSE.txt
MAINTAINERS.txt
misc/
modules/
phpinfo.php
scripts/
themes/
tiptoe.txt
update.php
xmlrpc.php
```

So the URL above will be requesting the root directory `/` of the Drupal site. Apache translates that into `index.php`. One variable/value pair is passed along with the request: the variable '`q`' is set to the value '`node/1`'.

So, let's pick up the show with the execution of `index.php`, which looks very simple and is only a few lines long.

Let's take a broad look at what happens during the execution of `index.php`. First, the `includes/bootstrap.inc` file is included, bringing in all the functions that are necessary to get Drupal's machinery up and running. There's a call to `drupal_page_header()`, which starts a timer, sets up caching, and notifies interested modules that the request is beginning. Next, the

`includes/common.inc` file is included, giving access to a wide variety of utility functions such as path formatting functions, form generation and validation, etc. The call to `fix_gpc_magic()` is there to check on the status of PHP "magic quotes" and to ensure that all escaped quotes enter Drupal's database consistently. Drupal then builds its navigation menu and sets the variable `$status` to the result of that operation. In the switch statement, Drupal checks for cases in which a Not Found or Access Denied message needs to be generated, and finally a call to `drupal_page_footer()`, which notifies all interested modules that the request is ending. Drupal closes up shop and the page is served. Simple, eh?

Let's delve a little more deeply into the process outlined above.

The first line of `index.php` includes the `includes/bootstrap.inc` file, but it also executes code towards the end of `bootstrap.inc`. First, it destroys any previous variable named `$conf`. Next, it calls `conf_init()`. This function allows Drupal to use site-specific configuration files, if it finds them. The name of the site-specific configuration file is based on the hostname of the server, as reported by PHP. `conf_init` returns the name of the site-specific configuration file; if no site-specific configuration file is found, sets the variable `$config` equal to the string `$confdir/default`. Next, it includes the named configuration file. Thus, in the default case it will include `sites/default/settings.php`. The code in `conf_init()` would be easier to understand if the variable `$file` were instead called `$potential_filename`. Likewise `$conf_filename` would be a better choice than `$config`.

The selected configuration file (normally `/sites/default/settings.php`) is now parsed, setting the `$db_url` variable, the optional `$db_prefix` variable, the `$base_url` for the website, and the `$languages` array (default is "`en"=>"english"`).

The `database.inc` file is now parsed, with the primary goal of initializing a connection to the database. If MySQL is being used, the `database.mysql.inc` files is brought in. Although the global variables `$db_prefix`, `$db_type`, and `$db_url` are set, the most useful result of parsing `database.inc` is a global variable called `$active_db` which contains the database connection handle.

Now that the database connection is set up, it's time to start a session by including the `includes/session.inc` file. Oddly, in this include file the executable code is located at the top of the file instead of the bottom. What the code does is to tell PHP to use Drupal's own session storage functions (located in this file) instead of the default PHP session code. A call to PHP's `session_start()` function thus calls Drupal's `sess_open()` and `sess_read()` functions. The `sess_read()` function creates a global `$user` object and sets the `$user->roles` array appropriately. Since I am running as an anonymous user, the `$user->roles` array contains one entry, `1->"anonymous user"`.

We have a database connection, a session has been set up...now it's time to get things set up for modules. The `includes/module.inc` file is included but no actual code is executed.

The last thing `bootstrap.inc` does is to set up the global variable `$conf`, an array of configuration options. It does this by calling the `variable_init()` function. If a per-site configuration file exists and has already populated the `$conf` variable, this populated array is passed in to `variable_init()`. Otherwise, the `$conf` variable is null and an empty array is passed in. In both cases, a populated array of name-value pairs is returned and assigned to the global `$conf` variable, where it will live for the duration of this request. It should be noted that name-value pairs in the per-site configuration file have precedence over name-value pairs retrieved from the "variable" table by `variable_init()`.

We're done with `bootstrap.inc`! Now it's time to go back to `index.php` and call `drupal_page_header()`. This function has two responsibilities. First, it starts a timer if `$conf['dev_timer']` is set; that is, if you are keeping track of page execution times. Second, if caching has been enabled it retrieves the cached page, calls `module_invoke_all()` for the 'init' and 'exit' hooks, and exits. If caching is not enabled or the page is not being served to an anonymous user (or several other special cases, like when feedback needs to be sent to a user), it simply exits and returns control to `index.php`.

Back at `index.php`, we find an include statement for `common.inc`. This file is chock-full of miscellaneous utility goodness, all kept in one file for performance reasons. But in addition to putting all these utility functions into our namespace, `common.inc` includes some files on its own. They include `theme.inc`, for theme support; `pager.inc` for paging through large datasets (it has nothing to do with calling your pager); and `menu.inc`. In `menu.inc`, many constants are defined that are used later by the menu system.

The next inclusion that `common.inc` makes is `xmlrpc.inc`, with all sorts of functions for dealing with XML-RPC calls. Although one would expect a quick check of whether or not this request is actually an XML-RPC call, no such check is done here. Instead, over 30 variable assignments are made, apparently so that if this request turns to actually be an XML-RPC call, they will be ready. An `xmlrpc_init()` function instead may help performance here?

A small `tablesort.inc` file is included as well, containing functions that help behind the scenes with sortable tables. Given the paucity of code here, a performance boost could be gained by moving these into `common.inc` itself.

The last include done by `common.inc` is `file.inc`, which contains common file handling functions. The constants `FILE_DOWNLOADS_PUBLIC = 1` and `FILE_DOWNLOADS_PRIVATE = 2` are set here, as well as the `FILE_SEPARATOR`, which is `\` for Windows machines and `/` for all others.

Finally, with includes finished, common.inc sets PHP's error handler to the `error_handler()` function in the common.inc file. This error handler creates a watchdog entry to record the error and, if any error reporting is enabled via the `error_reporting` directive in PHP's configuration file (`php.ini<code>`), it prints the error message to the screen. Drupal's `<code>error_handler()` does not use the last parameter `$variables`, which is an array that points to the active symbol table at the point the error occurred. The comment `// set error handler:`" at the end of common.inc is redundant, as it is readily apparent what the function call to `set_error_handler()` does.

The Content-Type header is now sent to the browser as a hard coded string: "Content-Type: text/html; charset=utf-8".

If you remember that the URL we are serving ends with `/~vandyk/drupal/?q=node/1`, you'll note that the variable `q` has been set. Drupal now parses this out and checks for any path aliasing for the value of `q`. If the value of `q` is a path alias, Drupal replaces the value of `q` with the actual path that the value of `q` is aliased to. This sleight-of-hand happens before any modules see the value of `q`. Cool.

Module initialization now happens via the `module_init()` function. This function runs `<code>require_once()` on the `<code>admin, filter, system, user and watchdog` modules. The filter module defines `FILTER_HTML*` and `FILTER_STYLE*` constants while being included. Next, other modules are `include_once'd` via `module_list()`. In order to be loaded, a module must (1) be enabled (that is, the status column of the "system" database table must be set to 1), and (2) Drupal's throttle mechanism must determine whether or not the module is eligible for exclusion when load is high. First, it determines whether the module is eligible by looking at the `throttle` column of the "system" database table; then, if the module is eligible, it looks at `$conf["throttle_level"]` to see whether the load is high enough to exclude the module. Once all modules have been `include_once'd` and their names added to the `$list` local array, the array is sorted by module name and returned. The returned `$list` is discarded because the `module_list()` invocation is not part of an assignment (e.g., it is simply `module_list()` and not `$module_list = module_list()`). The strategy here is to keep the module list inside a static variable called `$list` inside the `module_list()` function. The next time `module_list()` is called, it will simply return its static variable `$list` rather than rebuilding the whole array. We see that as we follow the final objective of `module_init()`; that is, to send all modules the "init" callback.

To see how the callbacks work let's step through the init callback for the first module. First `module_invoke_all()` is called and passed the string enumerating which callback is to be called. This string could be anything; it is simply a symbol that call modules have agreed to abide by, by convention. In this case it is the string "init".

The `module_invoke_all()` function now steps through the list of modules it got from calling `module_list()`. The first one is "admin", so it calls `module_invoke("admin", "init")`. The `module_invoke()` function simply puts the two together to get the name of the function it will call. In this case the name of the function to call is "`admin_init()`". If a function by this name exists, the function is called and the returned result, if any, ends up in an array called `$return` which is returned after all modules have been invoked. The lesson learned here is that if you are writing a module and intend to return a value from a callback, you must return it as an array. [Jonathan Chaffer: *Each "hook" (our word for what you call a callback) defines its own return type. See the full list of hooks available to module developers, with documentation about what they are expected to return.*]

Back to `common.inc`. There is a check for suspicious input data. To find out whether or not the user has permission to bypass this check, `user_access()` is called. This retrieves the user's permissions and stashes them in a static variable called `$perm`. Whether or not a user has permission for a given action is determined by a simple substring search for the name of the permission (e.g., "bypass input data check") within the `$perm` string. Our `$perm` string, as an anonymous user, is currently "0access content, ". Why the 0 at the beginning of the string? Because `$perm` is initialized to 0 by `user_access()`.

The actual check for suspicious input data is carried out by `valid_input_data()` which lives in `common.inc`. It simply goes through an array it's been handed (in this case the `$_REQUEST` array) and checks all keys and values for the following "evil" strings: javascript, expression, alert, dynsrc, datasrc, data, lowsrc, applet, script, object, style, embed, form, blink, meta, html, frame, iframe, layer, ilayer, head, frameset, xml. If any of these are matched watchdog records a warning and Drupal dies (in the PHP sense). I wondered why both the keys and values of the `$_REQUEST` array are examined. This seems very time-consuming. Also, would it die if my URL ended with "/?xml=true" or "/?format=xml"?

The next step in `common.inc`'s executable code is a call to `locale_init()` to set up locale data. If the user is not an anonymous user and has a language preference set up, the two-character language key is returned; otherwise, the key of the single-entry global array `$language` is returned. In our case, that's "en".

The last gasp of `common.inc` is to call `init_theme()`. You'd think that for consistency this would be called `theme_init()` (of course, that would be a namespace clash with a callback of the same name). This finds out which themes are available, which the user has selected, and then `include_once`'s the chosen theme. If the user's selected theme is not available, the value at `$conf["theme_default"]` is used. In our case, we are an anonymous user with no theme selected, so the default `xtemplate` theme is used. Thus, the file `themes/xtemplate/xtemplate.theme` is `include_once`'d. The inclusion of `xtemplate.theme` calls `include_once("themes/xtemplate/xtemplate.inc")`, and creates a new object called `xtemplate` as a global variable. Inside this object is an `xtemplate` object called "template" with lots of attributes. Then there is a nonfunctional line where `SetNullBlock` is called. A comment indicates that someone is aware that this

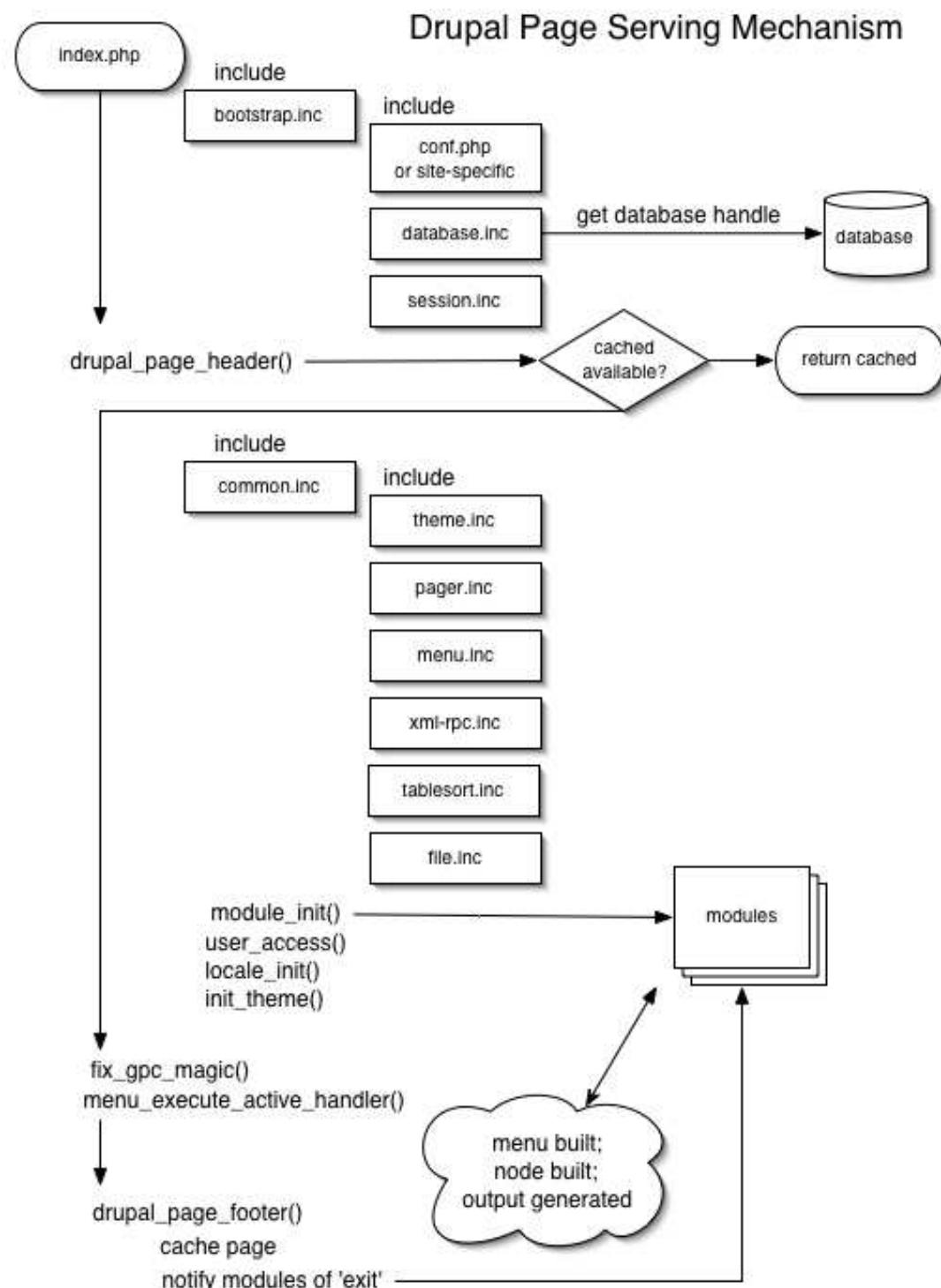
doesn't work.

Now we're back to `index.php`! A call to `fix_gpc_magic()` is in order. The "gpc" stands for Get, Post, Cookie: the three places that unescaped quotes may be found. If deemed necessary by the status of the boolean `magic_quotes_gpc` directive in PHP's configuration file (`php.ini`), slashes will be stripped from `$_GET`, `$_POST`, `$_COOKIE`, and `$_REQUEST` arrays. It seems odd that the function is not called `fix_gpc_magic_quotes`, since it is the "magic quotes" that are being fixed, not the magic. In my distribution of PHP, the `magic_quotes_gpc` directive is set to "Off", so slashes do not need to be stripped.

The next step is to set up menus. This step is crucial. The menu system doesn't just handle displaying menus to the user, but also determines what function will be handed the responsibility of displaying the page. The "q" variable (we usually call the Drupal path) is matched against the available menu items to find the appropriate callback to use. Much more information on this topic is available in the menu system documentation for developers. We jump to `menu_execute_active_handler()` in `menu.inc`. This sets up a `$_menu` array consisting of items, local tasks, path index, and visible arrays. Then the system realizes that we're not going to be building any menus for an anonymous user and bows out. The real meat of the node creation and formatting happens here, but is complex enough for a separate commentary. Back in `index.php`, the switch statement doesn't match either case and we approach the last call in the file, to `drupal_page_footer` in `common.inc`. This takes care of caching the page we've built if caching is enabled (it's not) and calls `module_invoke_all()` with the "exit" callback symbol.

Although you may think we're done, PHP's session handler still needs to tidy up. It calls `sess_write()` in `session.inc` to update the session database table, then `sess_close()` which simply returns 1.

We're done.



by John VanDyk

Creating modules - a tutorial

This tutorial describes how to create a module for Drupal 4.5.*. It is an update to the tutorial for Drupal 4.3. Please see comments there, also.

A module is a collection of functions that link into Drupal, providing additional functionality to your Drupal installation. After reading this tutorial, you will be able to create a basic block module and use it as a template for more advanced modules and node modules.

This tutorial will not necessarily prepare you to write modules for release into the wild. It does not cover caching, nor does it elaborate on permissions or security issues. Use this tutorial as a starting point, and review other modules and the Drupal handbook and Coding standards for more information.

This tutorial assumes the following about you:

- Basic PHP knowledge, including syntax and the concept of PHP objects
- Basic understanding of database tables, fields, records and SQL statements
- A working Drupal installation
- Drupal administration access
- Webserver access

This tutorial does not assume you have any knowledge about the inner workings of a Drupal module. This tutorial will not help you write modules for versions of Drupal earlier than 4.5.

Getting started

To focus this tutorial, we'll start by creating a block module that lists links to content such as blog entries or forum discussions that were created one week ago. The full tutorial will teach us how to create block content, write links, and retrieve information from Drupal nodes.

Start your module by creating a PHP file and save it as 'onthisdate.module' in the modules directory of your Drupal installation.

```
<?php  
?>
```

As per the Coding standards, use the longhand <?php tag, and not <? to enclose your PHP code.

All functions in your module are named {modulename}_{hook}, where "hook" is a well defined function name. Drupal will call these functions to get specific data, so having these well defined names means Drupal knows where to look.

Letting Drupal know about the new function

As mentioned above, the function we just wrote isn't a 'hook': it's not a Drupal recognized name. We need to tell Drupal how to access the function when displaying a page. We do this with the menu() hook. The menu() hook defines the association between a URL and the function that creates the content for that url. The hook also does permission checking, if desired.

```
<?php
function onthisdate_menu() {
  $items = array();
  $items[ ] = array('path' => 'onthisdate',
                    'callback' =>
                    '_onthisdate_all',
                    'access' =>
                    user_access('access content'),
                    'type' => MENU_CALLBACK);
  return $items;
}
?>
```

Basically, we're saying if the user goes to "onthisdate" (either via ?q=onthisdate or http://.../onthisdate), the content generated by onthisdate_all will be displayed. The title of the page will be "on this date". The type MENU_CALLBACK tells Drupal to not display the link in the user's menu, just use this function when the URL is accessed. Use MENU_LOCAL_TASK if you want the user to see the link in the side navigation block.

Navigate to /onthisdate (or ?q=onthisdate) and see what you get.

Telling Drupal about your module

The first function we'll write will tell Drupal information about your module: its name and description. The hook name for this function is 'help', so start with the onthisdate_help function:

```
<?php
function onthisdate_help($section='') {
}
?>
```

The \$section variable provides context for the help: where in Drupal or the module are we looking for help. The recommended way to process this variable is with a switch statement. You'll see this code pattern in other modules.

```
<?php
/**
 * Display help and module information
```

```

 * @param section which section of the site we're displaying
 * help
 * @return help text for section
 */
function onthisdate_help($section='') {
  $output = '';
  switch ($section) {
    case "admin/modules#description":
      $output = t("Displays links to nodes created on this
date");
      break;
  }
  return $output;
} // function onthisdate_help
?>
```

You will eventually want to add other cases to this switch statement to provide real help messages to the user. In particular, output for "admin/help#onthisdate" will display on the main help page accessed by the admin/help URL for this module (/admin/help or ?q=admin/help).

Telling Drupal who can use your module

The next function to write is the permissions function. The permissions function doesn't grant permission, it just specifies what permissions are available for this module. Access based on these permissions is defined later in the {module}_access function below. At this point, we'll give permission to anyone who can access site content or administrate the module:

```

<?php
/**
 * Valid permissions for this module
 * @return array An array of valid permissions for the
onthisdate module
*/
function onthisdate_perm() {
  return array('access content');
} // function onthisdate_perm()
?>
```

Conversely, if you are going to write a module that needs to have finer control over the permissions, and you're going to do permission control, you should expand this permission set. You can do this by adding strings to the array that is returned. For example:

```

<?php
function onthisdate_perm() {
  return array('access content', 'access onthisdate',
'administer onthisdate');
} // function onthisdate_perm
```

?>

For this tutorial, start with the first one. We'll later move to the second version.

You'll need to adjust who has permission to view your module on the administer Â» accounts Â» permissions page. We'll use the user_access function to check access permissions later (whoa, so many "laters!").

Your permission strings must be unique within your module. If they are not, the permissions page will list the same permission multiple times. They should also contain your module name, to avoid name space conflicts with other modules.

Announce we have block content

There are several types of modules: block modules and node modules are two. Block modules create abbreviated content that is typically (but not always, and not required to be) displayed along the left or right side of a page. Node modules generate full page content (such as blog, forum, or book pages).

We'll create a block content to start, and later discuss node content. A module can generate content for blocks and also for a full page (the blogs module is a good example of this). The hook for a block module is appropriately called "block", so let's start our next function:

```
<?php
/**
 * Generate HTML for the onthisdate block
 * @param op the operation from the URL
 * @param delta offset
 * @returns block HTML
 */
function onthisdate_block($op='list', $delta=0) {
} // end function onthisdate_block
?>
```

The block function takes two parameters: the operation and the offset, or delta. We'll just worry about the operation at this point. In particular, we care about the specific case where the block is being listed in the blocks page. In all other situations, we'll display the block content.

When the module will be listed on the blocks page, the \$op parameter's value will be 'list':

```
<?php
/**
 * Generate HTML for the onthisdate block
 * @param op the operation from the URL
 * @param delta offset
 * @returns block HTML
```

```

*/
function onthisdate_block($op='list', $delta=0) {
    // listing of blocks, such as on the admin/block page
    if ($op == "list") {
        $block[0]["info"] = t('On This Date');
        return $block;
    } else {
        // our block content
    }
} // end onthisdate_block
?>

```

Generate content for a block

Now, we need to generate the 'onthisdate' content for the block. Here we'll demonstrate a basic way to access the database.

Our goal is to get a list of content (stored as "nodes" in the database) created a week ago. Specifically, we want the content created between midnight and 11:59pm on the day one week ago. When a node is first created, the time of creation is stored in the database. We'll use this database field to find our data.

First, we need to calculate the time (in seconds since epoch start, see <http://www.php.net/manual/en/function.time.php> for more information on time format) for midnight a week ago, and 11:59pm a week ago. This part of the code is Drupal independent, see the PHP website (<http://php.net/>) for more details.

```

<?php
/**
 * Generate HTML for the onthisdate block
 * @param op the operation from the URL
 * @param delta offset
 * @returns block HTML
 */
function onthisdate_block($op='list', $delta=0) {
    // listing of blocks, such as on the admin/block page
    if ($op == "list") {
        $block[0]["info"] = t('On This Date');
        return $block;
    } else {
        // our block content
        // Get today's date
        $today = getdate();
        // calculate midnight one week ago
        $start_time = mktime(0, 0, 0,
                            $today['mon'], ($today['mday'] - 7),
                            $today['year']);

```

```

        // we want items that occur only on the day in question,
so
        // calculate 1 day
$end_time = $start_time + 86400;
// 60 * 60 * 24 = 86400 seconds in a day
...
}
?

```

The next step is the SQL statement that will retrieve the content we'd like to display from the database. We're selecting content from the node table, which is the central table for Drupal content. We'll get all sorts of content type with this query: blog entries, forum posts, etc. For this tutorial, this is okay. For a real module, you would adjust the SQL statement to select specific types of content (by adding the 'type' column and a WHERE clause checking the 'type' column).

Note: the table name is enclosed in curly braces: {node}. This is necessary so that your module will support database table name prefixes. You can find more information on the Drupal website by reading the Table Prefix (and sharing tables across instances) page in the Drupal handbook.

```

<?php
$query = "SELECT nid, title, created FROM ".
    "{node} WHERE created >= '" . $start_time .
    "' AND created <= '". $end_time . "'";
?

```

Drupal uses database helper functions to perform database queries. This means that, for the most part, you can write your database SQL statement and not worry about the backend connections.

We'll use db_query() to get the records (i.e. the database rows) that match our SQL query, and db_fetch_object() to look at the individual records:

```

<?php
// get the links
$queryResult = db_query($query);
// content variable that will be returned for display
$block_content = '';
while ($links = db_fetch_object($queryResult)) {
    $block_content .= '<a href="' . url('node/' .
$links->nid) . '">' .
                    $links->title . '</a><br />';
}
// check to see if there was any content before setting up
// the block
if ($block_content == '') {

```

```

        /* No content from a week ago. If we return nothing, the
block
         * doesn't show, which is what we want. */
        return;
    }
    // set up the block
    $block['subject'] = 'On This Date';
    $block['content'] = $block_content;
    return $block;
}
?>

```

Notice the actual URL is enclosed in the l() function. l generates links, adjust the URL to the installation's URL configuration of either clean URLs: http://(sitename)/node/2 or http://(sitename)/?q=node/2

Also, we return an array that has 'subject' and 'content' elements. This is what Drupal expects from a block function. If you do not include both of these, the block will not render properly.

You may also notice the bad coding practice of combining content with layout. If you are writing a module for others to use, you will want to provide an easy way for others (in particular, non-programmers) to adjust the content's layout. An easy way to do this is to include a class attribute in your link, or surround the HTML with a <div> tag with a module specific CSS class and not necessarily include the
 at the end of the link. Let's ignore this for now, but be aware of this issue when writing modules that others will use.

Putting it all together, our block function at this point looks like this:

```

<?php
function onthisdate_block($op='list', $delta=0) {
    // listing of blocks, such as on the admin/block page
    if ($op == "list") {
        $block[0]["info"] = t("On This Date");
        return $block;
    } else {
        // our block content
        // content variable that will be returned for display
        $block_content = '';
        // Get today's date
        $today = getdate();
        // calculate midnight one week ago
        $start_time = mktime(0, 0, 0, $today['mon'],
                             ($today['mday'] - 7),
                             $today['year']);
        // we want items that occur only on the day in question,
        so
        //calculate 1 day
    }
}
?>

```

```

$end_time = $start_time + 86400;
// 60 * 60 * 24 = 86400 seconds in a day
$query = "SELECT nid, title, created FROM ".
    "{node} WHERE created >= '" . $start_time .
    "' AND created <= '". $end_time . "'";
// get the links
$queryResult = db_query($query);
while ($links = db_fetch_object($queryResult)) {
    $block_content .= '<a
href="'.url('node/'.$links->nid).'">'.
    $links->title . '</a><br />';
}
// check to see if there was any content before setting
up the block
if ($block_content == '') {
    // no content from a week ago, return nothing.
    return;
}
// set up the block
$block['subject'] = 'On This Date';
$block['content'] = $block_content;
return $block;
}
?>

```

Installing, enabling and testing the module

At this point, you can install your module and it'll work. Let's do that, and see where we need to improve the module.

To install the module, you'll need to copy your `onthisdate.module` file to the `modules` directory of your Drupal installation. The file must be installed in this directory or a subdirectory of the `modules` directory, and must have the `.module` name extension.

Log in as your site administrator, and navigate to the modules administration page to get an alphabetical list of modules. In the menus: administer » modules, or via URL:

`http://.../admin/modules` or
`http://.../?q=admin/modules`

When you scroll down, you'll see the `onthisdate` module listed with the description next to it.

Enable the module by selecting the checkbox and save your configuration.

Because the module is a blocks module, we'll need to also enable it in the blocks administration menu and specify a location for it to display. Node modules may or may not need further configuration depending on the module. Any module can have settings, which affect the functionality/display of a module. We'll discuss settings later. For now, navigate to the blocks administration page: admin/block or administer » blocks in the menus.

Enable the module by selecting the enabled checkbox for the 'On This Date' block and save your blocks. Be sure to adjust the location (left/right) if you are using a theme that limits where blocks are displayed.

Now, head to another page, say, select the modules menu. In some themes, the blocks are displayed after the page has rendered the content, and you won't see the change until you go to new page.

If you have content that was created a week ago, the block will display with links to the content. If you don't have content, you'll need to fake some data. You can do this by creating a blog, forum topic or book page, and adjust the "Authored on:" date to be a week ago.

Alternately, if your site has been around for a while, you may have a lot of content created on the day one week ago, and you'll see a large number of links in the block.

Create a module configuration (settings) page

Now that we have a working module, we'd like to make it better. If we have a site that has been around for a while, content from a week ago might not be as interesting as content from a year ago. Similarly, if we have a busy site, we might not want to display all the links to content created last week. So, let's create a configuration page for the administrator to adjust this information.

A module's configuration is set up with the 'settings' hook. We would like only administrators to be able to access this page, so we'll do our first permissions check of the module here:

```
<?php
/**
 * Module configuration settings
 * @return settings HTML or deny access
 */
function onthisdate_settings() {
    // only administrators can access this module
    if (!user_access("admin_onthisdate")) {
        return message_access();
    }
}
?>
```

If you want to tie your module's permissions to the permissions of another module, you can use that module's permission string. The "access content" permission is a good one to check if the user can view the content on your site:

```
<?php
...
// check the user has content access
if (!user_access("access content")) {
    return message_access();
}
...
?>
```

We'd like to configure how many links display in the block, so we'll create a form for the administrator to set the number of links:

```
<?php
function onthisdate_settings() {
    // only administrators can access this module
    if (!user_access("admin onthisdate")) {
        return message_access();
    }
    $output .= form_textfield(t("Maximum number of links"),
"onthisdate_maxdisp",
variable_get("onthisdate_maxdisp", "3"), 2, 2,
t("The maximum number of links to display in the
block."));  return $output;
?>
```

This function uses several powerful Drupal form handling features. We don't need to worry about creating an HTML text field or the form, as Drupal will do so for us. We use variable_get to retrieve the value of the system configuration variable "onthisdate_maxdisp", which has a default value of 3. We use the form_textfield function to create the form and a text box of size 2, accepting a maximum length of 2 characters. We also use the translate function of t(). There are other form functions that will automatically create the HTML form elements for use. For now, we'll just use the form_textfield function.

Of course, we'll need to use the configuration value in our SQL SELECT, so we'll need to adjust our query statement in the onthisdate_block function:

```
<?php
$limitnum = variable_get("onthisdate_maxdisp", 3);
$query = "SELECT nid, title, created FROM ".
" {node} WHERE created >= '" . $start_time .
"' AND created <= '" . $end_time . "' LIMIT " .
$limitnum;
?>
```

You can test the settings page by editing the number of links displayed and noticing the block content adjusts accordingly.

Navigate to the settings page: admin/modules/onthisdate or administer » configuration » modules » onthisdate. Adjust the number of links and save the configuration. Notice the number of links in the block adjusts accordingly.

Note: We don't have any validation with this input. If you enter "c" in the maximum number of links, you'll break the block.

Adding menu links and creating page content

So far we have our working block and a settings page. The block displays a maximum number of links. However, there may be more links than the maximum we show. So, let's create a page that lists all the content that was created a week ago.

```
<?php
function onthisdate_all() {}?
?>
```

We're going to use much of the code from the block function. We'll write this ExtremeProgramming style, and duplicate the code. If we need to use it in a third place, we'll refactor it into a separate function. For now, copy the code to the new function _onthisdate_all(). Contrary to all our other functions, 'all', in this case, is not a Drupal hook. In our code, we can prefix this function with an underscore to help us remember this isn't a hook call. We'll discuss below.

```
<?php
function _onthisdate_all() {
    // content variable that will be returned for display
    $page_content = '';
    // Get today's date
    $today = getdate();
    // calculate midnight one week ago
    $start_time = mktime(0, 0, 0,
                           $today['mon'], ($today['mday'] - 1),
                           $today['year']);
    // we want items that occur only on the day in question,
    // so calculate 1 day
    $end_time = $start_time + 86400;
    // 60 * 60 * 24 = 86400 seconds in a day
    // NOTE! No LIMIT clause here! We want to show all the
    // code
    $query = "SELECT nid, title, created FROM ".
        "{node} WHERE created >= '" . $start_time .
        "' AND created <= '" . $end_time . "'";
    // get the links
    $queryResult = db_query($query);
```

```

while ($links = db_fetch_object($queryResult)) {
    $page_content .= '<a
href="'.url('node/'.$links->nid).'">'.
    $links->title . '</a><br />';
}
...
?

```

We have the page content at this point, but we want to do a little more with it than just return it. When creating pages, we need to send the page content to the theme for proper rendering. We use this with the theme() function. Themes control the look of a site. As noted above, we're including layout in the code. This is bad, and should be avoided. It is, however, the topic of another tutorial, so for now, we'll include the formatting in our content:

```

<?php
print theme("page", $content_string);
?>

```

The rest of our function checks to see if there is content and lets the user know. This is preferable to showing an empty or blank page, which may confuse the user.

Note that we are responsible for outputting the page content with the 'print theme()' syntax.

```

<?php
function _onthisdate_all() {
    ...
    // check to see if there was any content before
    // setting up the block
    if ($page_content == '') {
        // no content from a week ago, let the user know
        print theme("page",
                    "No events occurred on this site on this date
in history.");
        return;
    }
    print theme("page", $page_content);
}
?>

```

Adding a 'more' link and showing all entries

Because we have our function that creates a page with all the content created a week ago, we can link to it from the block with a "more" link.

Add these lines just before that `$block['subject']` line, adding this to the `$block_content` variable before saving it to the `$block['content']` variable:

```
<?php
// add a more link to our page that displays all the links
$block_content .=
    "<div class=\"more-link\">.
     l(t(\"more\"), \"onthisdate\", array(\"title\" => t(\"More
events on this day.\")))
    .\"</div>";
?>
```

This will add the more link.

Conclusion

We now have a working module. It created a block and a page. You should now have enough to get started writing your own modules. We recommend you start with a block module of your own and move onto a node module. Alternately, you can write a filter or theme.

As is, this tutorial's module isn't very useful. However, with a few enhancements, it can be entertaining. Try modifying the select query statement to select only nodes of type 'blog' and see what you get. Alternately, you could get only a particular user's content for a specific week. Instead of using the block function, consider expanding the menu and page functions, adding menus to specific entries or dates, or using the menu callback arguments to adjust what year you look at the content from.

If you start writing modules for others to use, you'll want to provide more details in your code. Comments in the code are incredibly valuable for other developers and users in understanding what's going on in your module. You'll also want to expand the help function, providing better help for the user. Follow the Drupal [Coding standards], especially if you're going to add your module to the project.

Two topics very important in module development are writing themeable pages and writing translatable content. Please check the Drupal Handbook for more details on these two subjects.

Updating your modules

As Drupal develops with each release it becomes necessary to update modules to take advantage of new features and stay functional with Drupal's API.

Converting 3.0 modules to 4.0

Converting modules from version 3.0 to version 4.0 standards requires rewriting the `form()` function, as follows:

Drupal 3.0:

```
function form($action, $form, $method = "post", $options = 0)
// Example
global $REQUEST_URI;
$form = form_hidden("nid", $nid);
print form($REQUEST_URI, $form);
```

Drupal 4.0:

```
function form($form, $method = "post", $action = 0, $options = 0)
// Example
$form = form_hidden("nid", $nid);
print form($form);
```

Converting 4.0 modules to 4.1

Drupal 4.1 changed the block hook function and taxonomy API. To convert a version 4.0 module to 4.1, the following changes must be made. First, the `*_block()` function must be re-written. Next, calls to `taxonomy_get_tree()` must be re-written to supply the parameters required by the new function. Finally, you may wish to take advantage of new functions added to the taxonomy API.

Required changes**Modified block hook:****Drupal 4.0:**

```
function *_block() {
  $blocks[0]["info"] = "First block info";
  $blocks[0]["subject"] = "First block subject";
  $blocks[0]["content"] = "First block content";
  $blocks[1]["info"] = "Second block info";
  $blocks[1]["subject"] = "Second block subject";
  $blocks[1]["content"] = "Second block content";
  // return array of blocks
  return $blocks;
}
```

Drupal 4.1:

```
function *_block($op = "list", $delta = 0) {
  if ($op == "list") {
    $blocks[0]["info"] = "First block info";
    $blocks[1]["info"] = "Second block info";
    return $blocks; // return array of block infos
  }
  else {
    switch($delta) {
      case 0:
        $block["subject"] = "First block subject";
        $block["content"] = "First block content";
        return $block;
      case 1:
        $block["subject"] = "Second block subject";
        $block["content"] = "Second block content";
    }
  }
}
```

```
        return $block;
    }
}
}
```

Modified taxonomy API:

Changes: in function taxonomy_get_tree()

- there is no longer a "parent" property; rather "parents" is an array
- the result tree is now returned instead of being passed by reference

Drupal 4.0:

```
function taxonomy_get_tree($vocabulary_id, &$tree, $parent = 0, $depth = -1, $key = "tid")
```

Drupal 4.1:

```
$tree = taxonomy_get_tree($vocabulary_id, $parents = 0, $depth = -1, $key = "tid")
```

Optional changes

- Take advantage of new taxonomy functions
taxonomy_get_vocabulary_by_name(\$name) and
taxonomy_get_term_by_name(\$name)
- Take advantage of pager functions
- Move hardcoded markup from modules to themes, using theme_invoke

Converting 4.1 modules to 4.2

Some points posted by Axel on drupal-devel on migrating 4.1.0 modules to CVS [updated and added to by ax]:

- the big "clean URL" patch: Over the weekend, [dries] bit the bullet and converted every single URL in Drupal's code. meaning we'll [can] have clean URLs like <http://foo.com/archive/2003/01/06>, <http://foo.com/user/42>, <http://foo.com/blog>, and so on.. meaning, for the code:
 - drupal_url(array("mod" => "search", "op" => "bla"),
"module"[, \$anchor = ""])
became
`url("search/bla"),`
with the first url part being the module, the second (typically) being the operation (\$op); more arguments are handled differently per module convention.
 - l("view node", array("op" => "view", "id" => \$nid),
"node"[, \$anchor = "", \$attributes = array()])
became
`l("view node", "node/view/$nid"[,$attributes =
array(), $query = NULL])`

- similar,
`lm()`, which meant "module link" and used to be
`module.php?mod=bla&op=blub...`, is now `l("title", "bla/blub/...")`; and
`la()`, which meant "admin link" and used to be
`admin.php?mod=bla&op=blub...`, is now `l("title", "admin/bla/blub/...")`
- After fixing those functions, you'll need to edit your `_page()` function and possibly others so that they get their arguments using the `arg()` function (see `includes/common.inc`). These arguments used to be globals called "mod", "op", "id" etc. now these same arguments must be accessed as `arg(1)`, `arg(3)`, for example.

- `$theme->function()` became `theme("function")`. see [drupal-devel] renaming 2 functions, [drupal-devel] `theme("function")` vs `$theme->function()` and [drupal-devel] [CVS] `theme()`
- `<module>_conf_options()` became
`<module>_settings()` - see [drupal-devel] renaming 2 functions.
note that doesn't get an extra menu entry, but
is accessed via "site configuration > modules > modules settings"
- the administration pages got changed quite a lot to use a "database driven link system" and become more logical/intuitive - see [drupal-devel] X-mas commit: administration pages. this first try resulted in poor performance and a not-so-good api, so it got refactored - see [PATCH] menus. this, as of time ax is writing this, isn't really satisfying, neither (you cannot build arbitrary menu-trees, some forms don't work (taxonomy > add term), ...), so it probably will change again. and i won't write more about this here.

well, this: you use `menu()` to add entries to the admin menu.
`menu("admin/node/nodes/0", "new or updated posts", "node_admin", "help", 0);` adds a menu entry "new or updated posts" 1 level below "post overview" (`admin/node/nodes`) and 2 level below "node management" (`admin/node`) (ie. at the 3. level), with a weight of 0 in the 3. level, with a line "help" below the main heading. for the callback ("node_admin") ... ask dries or zbynek

one more note, though: you do **not** add `<module>_settings()` to the menu (they automatically go to "site configuration > modules > module settings" - you only add `<module>_admin...()` ... things.

- [from `comment_is_new` function lost]

```
- comment_is_new($comment)
+ node_is_new($comment->nid, $comment->timestamp)
```

please add / update / correct!

Converting 4.2 modules to 4.3

Database table prefix

On 2003 Jul 10, Dries committed Slavica's table prefix patch which allows for a configurable "prefix to each drupal mysql table to easily share one database for multiply applications on server with only one database allowed." This patch requires all table names in SQL-queries to be enclosed in {curly brackets}, eg.

```
- db_query("DELETE FROM book WHERE nid = %d", $node->nid);
+ db_query("DELETE FROM {book} WHERE nid = %d", $node->nid);
```

so that the table prefix can be dynamically prepended to the table name. See the original feature request and the corresponding discussion at the mailing list for details.

New help system

From Michael Frankowski message:

There is a block of text placed at the top of each admin page by the `admin_page` function. After 4.3.0 is out the door the function `menu_get_active_help()` should probably be renamed/moved into the help module and be attached -- somehow -- to every `_page` hook (probably in the node module) so that we can use this system through out Drupal but for now, there is a block of text displayed at the top of every admin page. This is the active help block. (context sensitive help?)

If the URL of the admin page matches a URL in a `_help` hook then the text from that `_help` hook is displayed on the top of the admin page. If there is no match, the block is not displayed. Because Drupal matches URLs in order to stick "other" stuff in the `_help` hook we have taken to sticking descriptors after a "#" sign. So far, the following descriptors are recognised:

| Descriptor | Function |
|----------------------------------|---|
| admin/system/modules#name | The name of a module (unused, but there) |
| admin/system/modules#description | The description found on the admin/system/modules page. |
| admin/help#modulename | The module's help text, displayed on the admin/help page and through the module's individual help link. |
| user/help#modulename | The help for a distributed authorization module |

In the future we will probably recognise #block for the text needed in a block displayed by the help system.

Creating modules for version 4.3.1

This tutorial describes how to create a module for Drupal-CVS (i.e. Drupal version > 4.3.1). A module is a collection of functions that link into Drupal, providing additional functionality to your Drupal installation. After reading this tutorial, you will be able to create a basic block module and use it as a template for more advanced modules and node modules.

This tutorial will not necessarily prepare you to write modules for release into the wild. It does not cover caching, nor does it elaborate on permissions or security issues. Use this tutorial as a starting point, and review other modules and the [Drupal handbook] and [Coding standards] for more information.

This tutorial assumes the following about you:

- Basic PHP knowledge, including syntax and the concept of PHP objects
- Basic understanding of database tables, fields, records and SQL statements
- A working Drupal installation
- Drupal administration access
- Webserver access

This tutorial does not assume you have any knowledge about the inner workings of a Drupal module. This tutorial will not help you write modules for Drupal 4.3.1 or before.

Getting Started

To focus this tutorial, we'll start by creating a block module that lists links to content such as blog entries or forum discussions that were created one week ago. The full tutorial will teach us how to create block content, write links, and retrieve information from Drupal nodes.

Start your module by creating a PHP file and save it as 'onthisdate.module'.

```
<?php  
?>
```

As per the [Coding standards], use the longhand <?php tag, and not <? to enclose your PHP code.

All functions in your module are named {modulename}_{hook}, where "hook" is a well defined function name. Drupal will call these functions to get specific data, so having these well defined names means Drupal knows where to look.

Telling Drupal about your module

The first function we'll write will tell Drupal information about your module: its name and description. The hook name for this function is 'help', so start with the onthisdate_help function:

```
<?php
function onthisdate_help($section) {
}
?>
```

The \$section variable provides context for the help: where in Drupal or the module are we looking for help. The recommended way to process this variable is with a switch statement. You'll see this code pattern in other modules.

```
<?php
/* Commented out until bug fixed */
/*
function onthisdate_help($section) {
    switch($section) {
        case "admin/system/modules#name":
            $output = "onthisdate";
            break;
        case "admin/system/modules#description":
            $output = "Display a list of nodes that were created
a week ago.";
            break;
        default:
            $output = "onthisdate";
            break;
    }
    return $output;
}
*/
?>
```

You will eventually want to add other cases to this switch statement to provide real help messages to the user. In particular, output for "admin/help#onthisdate" will display on the main help page accessed by the admin/help URL for this module (/admin/help or ?q=admin/help).

Note:This function is commented out in the above code. This is on purpose, as the current version of Drupal CVS won't display the module name, and won't enable it properly when installed. Until this bug is fixed, comment out your help function, or your module may not work.

Telling Drupal who can use your module

The next function to write is the permissions function. Here, you can tell Drupal who can access your module. At this point, give permission to anyone who can access site content or administrate the module.

```
<?php
function onthisdate_perm() {
    return array("administer onthisdate");
}
?>
```

If you are going to write a module that needs to have finer control over the permissions, and you're going to do permission control, you may want to define a new permission set. You can do this by adding strings to the array that is returned:

```
<?php
function onthisdate_perm() {
    return array("access onthisdate", "administer
onthisdate");
}
?>
```

You'll need to adjust who has permission to view your module on the administer » accounts » permissions page. We'll use the user_access function to check access permissions later.

Be sure your permission strings must be unique to your module. If they are not, the permissions page will list the same permission multiple times.

Announce we have block content

There are several types of modules: block modules and node modules are two. Block modules create abbreviated content that is typically (but not always, and not required to be) displayed along the left or right side of a page. Node modules generate full page content (such as blog, forum, or book pages).

We'll create a block content to start, and later discuss node content. A module can generate content for blocks and also for a full page (the blogs module is a good example of this). The hook for a block module is appropriately called "block", so let's start our next function:

```
<?php
function onthisdate_block($op='list', $delta=0) {
}
?>
```

The block function takes two parameters: the operation and the offset, or delta. We'll just worry about the operation at this point. In particular, we care about the specific case where the block is being listed in the blocks page. In all other situations, we'll display the block content.

```
<?php
function onthisdate_block($op='list', $delta=0) {
    // listing of blocks, such as on the admin/system/block
    page
    if ($op == "list") {
        $block[0]["info"] = t("On This Date");
        return $block;
    } else {
        // our block content
    }
}
?>
```

Generate content for a block

Now, we need to generate the 'onthisdate' content for the block. In here, we'll demonstrate a basic way to access the database.

Our goal is to get a list of content (stored as "nodes" in the database) created a week ago. Specifically, we want the content created between midnight and 11:59pm on the day one week ago. When a node is first created, the time of creation is stored in the database. We'll use this database field to find our data.

First, we need to calculate the time (in seconds since epoch start, see <http://www.php.net/manual/en/function.time.php> for more information on time format) for midnight a week ago, and 11:59pm a week ago. This part of the code is Drupal independent, see the PHP website (<http://php.net/>) for more details.

```
<?php
function onthisdate_block($op='list', $delta=0) {
    // listing of blocks, such as on the admin/system/block
    page
    if ($op == "list") {
        $block[0]["info"] = t("On This Date");
        return $block;
    } else {
        // our block content
        // Get today's date
        $today = getdate();
        // calculate midnight one week ago
        $start_time = mktime(0, 0, 0,
            $today['mon'], ($today['mday']
    7), $today['year']);
            // we want items that occur only on the day in
        question, so calculate 1 day
        $end_time = $start_time + 86400; // 60 * 60 * 24 =
        86400 seconds in a day
        ...
    }
}
?>
```

The next step is the SQL statement that will retrieve the content we'd like to display from the database. We're selecting content from the node table, which is the central table for Drupal content. We'll get all sorts of content type with this query: blog entries, forum posts, etc. For this tutorial, this is okay. For a real module, you would adjust the SQL statement to select specific types of content (by adding the 'type' column and a WHERE clause checking the 'type' column).

Note: the table name is enclosed in curly braces: {node}. This is necessary so that your module will support database table name prefixes. You can find more information on the Drupal website by reading the [Table Prefix (and sharing tables across instances)] page in the Drupal handbook.

```
<?php
$query = "SELECT nid, title, created FROM " .
    "{node} WHERE created >= '" . $start_time .
```

```
"' AND created <= '" . $end_time . "'";  
?>
```

Drupal uses database helper functions to perform database queries. This means that, for the most part, you can write your database SQL statement and not worry about the backend connections.

We'll use db_query() to get the records (i.e. the database rows) that match our SQL query, and db_fetch_object() to look at the individual records:

```
<?php  
    // get the links  
    $queryResult = db_query($query);  
    // content variable that will be returned for display  
    $block_content = '';  
    while ($links = db_fetch_object($queryResult)) {  
        $block_content .= '<a href="' . url('node/view/' .  
$links->nid) . '">' .  
                           $links->title . '</a><br />;  
    }  
    // check to see if there was any content before setting  
    up the block  
    if ($block_content == '') {  
        /* No content from a week ago. If we return nothing,  
        the block  
        * doesn't show, which is what we want. */  
        return;  
    }  
    // set up the block  
    $block['subject'] = 'On This Date';  
    $block['content'] = $block_content;  
    return $block;  
}  
?>
```

Notice the actual URL is enclosed in the url() function. This adjusts the URL to the installations URL configuration of either clean URLs: <http://sitename/node/view/2> or <http://sitename/?q=node/view/2>

Also, we return an array that has 'subject' and 'content' elements. This is what Drupal expects from a block function. If you do not include both of these, the block will not render properly.

You may also notice the bad coding practice of combining content with layout. If you are writing a module for others to use, you will want to provide an easy way for others (in particular, non-programmers) to adjust the content's layout. An easy way to do this is to include a

class attribute in your link, and not necessarily include the
 at the end of the link. Let's ignore this for now, but be aware of this issue when writing modules that others will use.

Putting it all together, our block function looks like this:

```
<?php
function onthisdate_block($op='list', $delta=0) {
    // listing of blocks, such as on the admin/system/block
    page
    if ($op == "list") {
        $block[0]["info"] = t("On This Date");
        return $block;
    } else {
        // our block content
        // content variable that will be returned for display
        $block_content = '';
        // Get today's date
        $today = getdate();
        // calculate midnight one week ago
        $start_time = mktime(0, 0, 0,
                                $today['mon'], ($today['mday'] - 7), $today['year']);
        // we want items that occur only on the day in
        question, so calculate 1 day
        $end_time = $start_time + 86400; // 60 * 60 * 24 =
        86400 seconds in a day
        $query = "SELECT nid, title, created FROM ".
            "{node} WHERE created >= '" . $start_time .
            "' AND created <= '". $end_time . "'";
        // get the links
        $queryResult = db_query($query);
        while ($links = db_fetch_object($queryResult)) {
            $block_content .= '<a
href="'.url('node/view/'.$links->nid).'">' .
                $links->title . '</a><br />';
        }
        // check to see if there was any content before
        setting up the block
        if ($block_content == '') {
            // no content from a week ago, return nothing.
            return;
        }
        // set up the block
        $block['subject'] = 'On This Date';
        $block['content'] = $block_content;
    }
    return $block;
}
```

```
    }  
}  
?>
```

Installing, enabling and testing the module

At this point, you can install your module and it'll work. Let's do that, and see where we need to improve the module.

To install the module, you'll need to copy your `onthisdate.module` file to the `modules` directory of your Drupal installation. The file must be installed in this directory or a subdirectory of the `modules` directory, and must have the `.module` name extension.

Log in as your site administrator, and navigate to the modules administration page to get an alphabetical list of modules. In the menus: administer → configuration → modules, or via URL:

`http://.../admin/system/modules` **or**
`http://.../?q=admin/system/modules`

Note: You'll see one of three things for the 'onthisdate' module at this point:

- You'll see the 'onthisdate' module name and no description
- You'll see no module name, but the 'onthisdate' description
- You'll see both the module name and the description

Which of these three choices you see is dependent on the state of the CVS tree, your installation and the help function in your module. If you have a description and no module name, and this bothers you, comment out the help function for the moment. You'll then have the module name, but no description. For this tutorial, either is okay, as you will just enable the module, and won't use the help system.

Enable the module by selecting the checkbox and save your configuration.

Because the module is a blocks module, we'll need to also enable it in the blocks administration menu and specify a location for it to display. Navigate to the blocks administration page: `admin/system/block` or administer → configuration → blocks in the menus.

Enable the module by selecting the enabled checkbox for the 'On This Date' block and save your blocks. Be sure to adjust the location (left/right) if you are using a theme that limits where blocks are displayed.

Now, head to another page, say select the module. In some themes, the blocks are displayed after the page has rendered the content, and you won't see the change until you go to new page.

If you have content that was created a week ago, the block will display with links to the content. If you don't have content, you'll need to fake some data. You can do this by creating a blog, forum topic or book page, and adjust the "Authored on:" date to be a week ago.

Alternately, if your site has been around for a while, you may have a lot of content created on the day one week ago, and you'll see a large number of links in the block.

Create a module configuration (settings) page

Now that we have a working module, we'd like to make it better. If we have a site that has been around for a while, content from a week ago might not be as interesting as content from a year ago. Similarly, if we have a busy site, we might not want to display all the links to content created last week. So, let's create a configuration page for the administrator to adjust this information.

The configuration page uses the 'settings' hook. We would like only administrators to be able to access this page, so we'll do our first permissions check of the module here:

```
<?php
function onthisdate_settings() {
    // only administrators can access this module
    if (!user_access("admin onthisdate")) {
        return message_access();
    }
}
?>
```

If you want to tie your modules permissions to the permissions of another module, you can use that module's permission string. The "access content" permission is a good one to check if the user can view the content on your site:

```
<?php
...
// check the user has content access
if (!user_access("access content")) {
    return message_access();
}
...
?>
```

We'd like to configure how many links display in the block, so we'll create a form for the administrator to set the number of links:

```
<?php
function onthisdate_settings() {
    // only administrators can access this module
    if (!user_access("admin onthisdate")) {
        return message_access();
    }
    $output .= form_textfield(t("Maximum number of links"),
    "onthisdate_maxdisp",
                                variable_get("onthisdate_maxdisp", "3"),
    2,
                                t("The maximum number of links to display in
the block."));
    return $output;
}
?>
```

This function uses several powerful Drupal form handling features. We don't need to worry about creating an HTML text field or the form, as Drupal will do so for us. We use `variable_get` to retrieve the value of the system configuration variable "onthisdate_maxdisp", which has a default value of 3. We use the `form_textfield` function to create the form and a text box of size 2, accepting a maximum length of 2 characters. We also use the `translate` function of `t()`. There are other form functions that will automatically create the HTML form elements for use. For now, we'll just use the `form_textfield` function.

Of course, we'll need to use the configuration value in our SQL SELECT, so we'll need to adjust our query statement in the `onthisdate_block` function:

```
<?php
$limitnum = variable_get("onthisdate_maxdisp", 3);
$query = "SELECT nid, title, created FROM ".
    "{node} WHERE created >= '" . $start_time .
    "' AND created <= '". $end_time . "' LIMIT " .
$limitnum;
?>
```

You can test the settings page by editing the number of links displayed and noticing the block content adjusts accordingly.

Navigate to the settings page: `admin/system/modules/onthisdate` or `administer » configuration » modules » onthisdate`. Adjust the number of links and save the configuration. Notice the number of links in the block adjusts accordingly.

Note: We don't have any validation with this input. If you enter "c" in the maximum number of links, you'll break the block.

Adding menu links and creating page content

So far we have our working block and a settings page. The block displays a maximum number of links. However, there may be more links than the maximum we show. So, let's create a page that lists all the content that was created a week ago.

```
<?php
function onthisdate_all() {
}
?>
```

We're going to use much of the code from the block function. We'll write this ExtremeProgramming style, and duplicate the code. If we need to use it in a third place, we'll refactor it into a separate function. For now, copy the code to the new function `onthisdate_all()`. Contrary to all our other functions, 'all', in this case, is not a Drupal hook. We'll discuss below.

```
<?php
function onthisdate_all() {
    // content variable that will be returned for display
    $page_content = '';
    // Get today's date
    $today = getdate();
    // calculate midnight one week ago
    $start_time = mktime(0, 0, 0,
                           $today['mon'
    7), $today['year']);
    // we want items that occur only on the day in question,
    // so calculate 1 day
    $end_time = $start_time + 86400;    // 60 * 60 * 24 =
    86400 seconds in a day
    // NOTE! No LIMIT clause here! We want to show all the
    // code
    $query = "SELECT nid, title, created FROM ".
        "{node} WHERE created >= '" . $start_time .
        "' AND created <= '". $end_time . "'";
    // get the links
    $queryResult = db_query($query);
    while ($links = db_fetch_object($queryResult)) {
        $page_content .= '<a
href="'.url('node/view/'.$links->nid).'">'.
            $links->title . '</a><br />';
```

```

    }
    ...
}
?>
```

We have the page content at this point, but we want to do a little more with it than just return it. When creating pages, we need to send the page content to the theme for proper rendering. We use this with the theme() function. Themes control the look of a site. As noted above, we're including layout in the code. This is bad, and should be avoided. It is, however, the topic of another tutorial, so for now, we'll include the formatting in our content:

```
<?php
    print theme("page", $content_string);
?>
```

The rest of our function checks to see if there is content and lets the user know. This is preferable to showing an empty or blank page, which may confuse the user.

Note that we are responsible for outputting the page content with the 'print theme()' syntax. This is a change from previous 4.3.x themes.

```
<?php
function onthisdate_all() {
    ...
    // check to see if there was any content before setting
    up the block
    if ($page_content == '') {
        // no content from a week ago, let the user know
        print theme("page",
                    "No events occurred on this site on this
date in history.");
        return;
    }
    print theme("page", $page_content);
}
?>
```

Letting Drupal know about the new function

As mentioned above, the function we just wrote isn't a 'hook': it's not a Drupal recognized name. We need to tell Drupal how to access the function when displaying a page. We do this with the _link hook and the menu() function:

```
<?php
function onthisdate_link($type, $node=0) {
}
?>
```

There are many different types, but we're going to use only 'system' in this tutorial.

```
<?php
function onthisdate_link($type, $node=0) {
  if (($type == "system")) {
    // URL, page title, func called for page content, arg,
    1 = don't disp menu
    menu("onthisdate", t("On This Date"),
"onthisdate_all", 1, 1);
  }
}
?>
```

Basically, we're saying if the user goes to "onthisdate" (either via ?q=onthisdate or http://.../onthisdate), the content generated by onthisdate_all will be displayed. The title of the page will be "On This Date". The final "1" in the arguments tells Drupal to not display the link in the user's menu. Make this "0" if you want the user to see the link in the side navigation block.

Navigate to /onthisdate (or ?q=onthisdate) and see what you get.

Adding a more link and showing all entries

Because we have our function that creates a page with all the content created a week ago, we can link to it from the block with a "more" link.

Add these lines just before that \$block['subject'] line, adding this to the \$block_content variable before saving it to the \$block['content'] variable:

```
<?php
  // add a more link to our page that displays all the
  links
  $block_content .= "<div class=\"more-link\">".
l(t("more"), "onthisdate", array("title" => t("More events
on this day."))) . "</div>";
?
?>
```

This will add the more link.

Conclusion

We now have a working module. It created a block and a page. You should now have enough to get started writing your own modules. We recommend you start with a block module of your own and move onto a node module. Alternately, you can write a filter or theme.

As is, this tutorial's module isn't very useful. However, with a few enhancements, it can be entertaining. Try modifying the select query statement to select only nodes of type 'blog' and see what you get. Alternately, you could get only a particular user's content for a specific week. Instead of using the block function, consider expanding the menu and page functions, adding menus to specific entries or dates, or using the menu callback arguments to adjust what year you look at the content from.

If you start writing modules for others to use, you'll want to provide more details in your code. Comments in the code are incredibly valuable for other developers and users in understanding what's going on in your module. You'll also want to expand the help function, providing better help for the user. Follow the Drupal [Coding standards], especially if you're going to add your module to the project.

Two topics very important in module development are writing themeable pages and writing translatable content. Please check the [Drupal Handbook] for more details on these two subject.

How to build up a _help hook

The following template can be used to build a _help hook.

```
<?php
function <modulename>_help($section) {
  $output = "";
  switch ($section) {
  }
  return $output;
}
?>
```

In the template replace *modulename* with the name of your module.

If you want to add help text to the overall administrative section. (admin/help) stick this inside the switch:

```
<?php
case 'admin/help#<modulename>':
  $output = t('The text you want displayed');
```

```
        break;
?>
```

If you also want this same text displayed for an individual help link in your menu area. You have this kind of tree:

```
+ Administration
|
-> Your area
| |
| -> Your configuration
| -> help
|
-> Overall admin help.
```

Change the function line to this:

```
<?php
function <modulename>_help($section =
'admin/help#<modulename>') {
?>
```

Now that you have the template started place a case statement in for any URL you want a "context sensitive" help message in the admin section. An example, you have a page that individually configures your module, it is at admin/system/modules/, you want to add some text to the top help area.

```
<?php
case 'admin/system/modules/<modulename>':
    $output = t('Your new help text');
    break;
?>
```

How to convert a _system hook

There are three things that can appear in a _system hook:

| Field | Function |
|--------------------------|---|
| \$field == "name" | The module name |
| \$field == "description" | The description placed in the module list |
| \$field == "admin-help" | The help text placed at the TOP of this module's individual configuration area. |

Take the text for each one and move it into the _help hook. Replace the \$system[<name>] that is normally at the front of each one with \$output, now place a "break;" after the line and a case '<name>': before it where

name is one of the following:

- If \$system is \$system["name"] then the case is case 'admin/system/modules#name'
- If \$system is \$system["description"] then case is case 'admin/system/modules#description'
- If \$system is \$system["admin-help"] then the case is case 'admin/system/modules/<modulename>'

Now remove the _system function and you are done.

An example:

```
<?php
function example_system($field){
    $system["description"] = t("This is my example _system
hook to convert for
the help system I have spent a lot of time with.");
    $system["admin-help"] = t("Can you believe that I would
actually write an
individual setup page on an EXAMPLE module??");
    return $system[$field];
}
?>
<?php
function example_help($section) {
    $output = "";
    switch ($section) {
        case 'admin/system/modules#example':
            $output = t("This is my example _system hook to
convert for the help
system I have spent a lot of time with.");
            break;
        case 'admin/system/modules/example':
            $output = t("Can you believe that I would actually
write an individual
setup page on an EXAMPLE module??");
            break;
    }
    return $output;
}
?>
```

How to convert an _auth_help hook

Okay, you have written your Distributed Authorization module, and given us a great help text for it and I had to go and ruin it all by changing the help system. What a terrible thing for me to do. How do you convert it?

It is not that hard. There are two places you have to deal with:

1. The text inside the _auth_help hook needs to be moved inside the _help hook under the section user/help#<modulename> and
2. You have to change the _page hook, which normally displays that help text, to find your text in a new location by changing the function call <modulename>_auth_help() to <modulename>_help("user/help#<modulename>").

See, it is not THAT terrible.

An example:

```
<?php
function exampled_a_page() {
    theme("header");
    theme("box", "Example DA", exampled_a_auth_help());
    theme("footer");
}
function exampled_a_auth_help() {
    $site = variable_get("site_name", "this web site");
    $html_output =
        <p>This is my example Distributed Auth help. Using this
example you cannot login to <i>%s</i> because it has no
_auth hook.&</p>
<p><u>BUT</u> you should still use Drupal since it is a
<b>GREAT</b> CMS and is only getting better.</p>
<p>To learn about about Drupal you can <a
href=\"www.drupal.org\">visit the site</a></p>;
    return sprintf(t($html_output), $site);
}
?>
<?php
function exampled_a_page() {
    theme("header");
    theme("box", "Example DA",
exampled_a_help('user/help#exampled_a'));
    theme("footer");
}
function exampled_a_help($section) {
    $output = "";
    if ($section == 'user/help#exampled_a') {
        $output = "This is my example Distributed Auth help. Using this
example you cannot login to <i>%s</i> because it has no
_auth hook. But you should still use Drupal since it is a
<b>GREAT</b> CMS and is only getting better. To learn about
about Drupal you can <a href='http://www.drupal.org'>visit the
site</a>.";
```

```

        switch ($section) {
            case 'user/help#examplepeda':
                $site = variable_get("site_name", "this web site");
                $output .= "<p>This is my example Distributed Auth
help. Using this example you cannot login to %site because it has no
_auth hook.</p>";
                $output .= "<p><u>BUT</u> you should still use
Drupal
since it is a <b>GREAT</b> CMS and is only getting
better.</p>";
                $output .= "<p>To learn about about Drupal you can
visit %drupal.</p>";
                $output = t($output, array("%site" =>
"<i>$site</i>",
"%drupal" => "<a href=\"www.drupal.org\">visit the
site</a>"));
                break;
        }
        return $output
    }
?>
```

Converting 4.3 modules to 4.4

Since Drupal 4.3, major changes have been made to the theme, menu, and node systems. Most themes and modules will require some changes.

Menu system

The Drupal menu system has been extended to drive all pages, not just administrative pages. This is continuing the work done for Drupal 4.3, which integrated the administrative menu with the user menu. We now have consistency between administrative and "normal" pages; when you learn to create one, you know how to create the other.

The flow of page generation now proceeds as follows:

1. The `_link` hook in all modules is called, so that modules can use `menu()` to add items to the menu. For example, a module could define:

```

<?php
function example_link($type) {
    if ($type == "system") {
        menu("example", t("example"), "example_page");
        menu("example/foo", t("foo"), "example_foo");
    }
}
?>
```

2. The menu system examines the current URL, and finds the "best fit" for the URL in the menu. For example, if the current URL is example/foo/bar/12, the above menu() calls would cause example_foo("bar", 12) to get invoked.
3. The callback may set the title or breadcrumb trail if the defaults are not satisfactory (more on this later).
4. The callback is responsible for printing the requested page. This will usually involve preparing the content, and then printing the return value of theme("page"). For example:

```
<?php
function example_foo($theString, $theNumber) {
  $output = $theString. " - " . $theNumber;
  print theme("page", $output);
}
?>
```

The following points should be considered when upgrading modules to use the new menu system:

- The _page hook is obsolete. Pages will not be shown unless they are declared with a menu() call as discussed above. To convert former _page hooks to the new system as simply as possible, just declare that function as a "catchall" callback:


```
<?php
menu("example", t("example"), "example_page", 0,
MENU_HIDE);
?>
```

 The trailing MENU_HIDE argument in this call makes the menu item hidden, so the callback functions but the module does not clutter the user menu.
- Old administrative callbacks returned their content. In the new system, administrative and normal callbacks alike are responsible for printing the entire page.
- The title of the page is printed by the theme system, so page content does not need to be wrapped in a theme("box") to get a title printed. If the default title is not satisfactory, it can be changed by calling drupal_set_title(\$title) before theme("page") gets called, or by passing the title to theme("page") as a parameter.
- The breadcrumb trail is also printed by the theme. If the default one needs to be overridden (to present things like forum hierarchies), this can be done by calling drupal_set_breadcrumb(\$breadcrumb) before theme("page") gets called, or by passing the breadcrumb to theme("page") as a parameter. \$breadcrumb should be a list of links beginning with "Home" and proceeding up to, but not including, the current page.

Theme system

For full information on theme system changes, see converting 4.3 themes to CVS. The following points are directly relevant to module development:

- All theme functions now return their output instead of printing them to the user. Old theme() usage:

```
<?php
theme("box", $title, $output);
?>
```

New usage:

```
<?php
print theme("box", $title, $output);
?>
```

Modules that define their own theme functions should also return their output.

- The naming of theme functions defined by modules has been standardized to theme_<module>_name;. When using a theme function there is no need to include the theme_ part, as theme() will do this automatically. Example:

```
<?php
function theme_example_list($list) {
    return implode('<br />', $list);
}
print theme('example_list', array(1,2,3));
?>
```

Theme functions must always be called using theme() to allow for the active theme to modify the output if necessary.

- The theme("header") and theme("footer") functions are not available anymore. Module developers should use the theme("page") function which wraps the content in the site theme. The full syntax of this function is

```
<?php
theme("page", $output, $title, $breadcrumb);
?>
```

where \$title and \$breadcrumb will override any values set before for these properties.

Node system

The node system has been upgraded to allow a single module to define more than one type of node. This will allow some of the more convoluted code in, for example, project.module to be tidied up.

- The _node() hook has been deprecated. In its place, modules that define nodes should use _node_name() and _help().
- The _node_name() function should return a translated string containing the human-readable name of the node type.

- The `_help()` function, when called with parameter "node/add#modulename", should return a translated string containing the description of the node type.
- Modules wishing to use the new ability to define multiple node types should see the Doxygen documentation for `hook_node_name()` and `hook_node_types()`.

Filter system

- The various filter hooks ('filter', 'conf_filters') have been merged into one 'filter' hook. A module that provides filtering functionality should implement:

```
<?php
function example_filter($op, $text = "") {
  switch ($op) {
    case "name":
      return t("Name of the filter");
    case "prepare":
      // Do preparing on $text
      return $text;
    case "process":
      // Do processing on $text
      return $text;
    case "settings":
      // Generate $output of settings
      return $output;
  }
}
?>
```

- "name" is new, and should return a friendly name for the filter.
- "prepare" is also new. This is an extra step that is performed before the default HTML processing, if HTML tags are allowed. It is meant to give filters the chance to escape HTML-like data before it can get stripped. This means, to convert meaningful HTML characters like < and > into entities such as < and >.

Common examples include filtering pieces of PHP code, mathematical formulas, etc. It is not allowed to do anything other than escaping in the "prepare" step.

If your filter currently performs such a step in the main "process" step, it should be moved into "prepare" instead. If you don't need any escaping, your filter should simply return \$text without processing in this case.

- "process" is the equivalent of the old "filter" hook. Normal filtering is performed here, and the changed \$text is returned.
- "settings" is the equivalent of the old "conf_filters" hook. If your filter

provides configurable options, you should return them here (using the standard `form_*` functions).

- The filter handling code has been moved to a new required `filter.module`, and thus most of the filter function names changed, although none of those should have been called from modules. The `check_output()` function is still available with the same functionality.
- Node filtering is optimized with the `node_prepare()` function now, which only runs the body through the filters if the node view page is displayed. Otherwise, only the teaser is filtered.
- The `_compose_tips` hook (defined by the contrib `compose_tips.module`) is not supported anymore, but more advanced functionality exists in the core. You can emit extensive compose tips related to the filter you define via the `_help` hook with the '`filter#long-tip`' section identifier. The `compose_tips` URL is thus changed to `filter/tips`. The `form_allowed_tags_text()` function is replaced with `filter_tips_short()`, which now supports short tips to be placed under textareas. Any module can inject short tips about the filter defined via the `_help` hook, with the '`filter#short-tip`' section identifier.

Hook changes

Other than those mentioned above, the following hooks have changed:

- The `_view` hook has been changed to return its content rather than printing it. It also has an extra parameter, `$page`, that indicates whether the node is being viewed as a standalone page or as part of a larger context. This is important because nodes may change the breadcrumb trail if they are being viewed as a page. Old usage:

```
<?php
function example_view($node, $main = 0) {
  if ($main) {
    theme("node", $node, $main);
  }
  else {
    $breadcrumb[ ] = l(t("Home"), "");
    $breadcrumb[ ] = l(t("foo"), "foo");
    $node->body = theme("breadcrumb", $breadcrumb) . "<br
/>" . $node->body;
    theme("node", $node, $main);
  }
}
?>
```

New usage:

```
<?php
function example_view($node, $main = 0, $page = 0) {
  if ($main) {
    return theme("node", $node, $main, $page);
```

```

    }
else {
  if ($page) {
    $breadcrumb[ ] = l(t("Home"), "");
    $breadcrumb[ ] = l(t("foo"), "foo");
    drupal_set_breadcrumb($breadcrumb);
  }
  return theme("node", $node, $main, $page);
}
?>

```

- The `_form` hook used by *node modules* no longer takes 3 arguments. The second argument `$help`, typically used to print submission guidelines, has been removed. Instead, the help should be emitted using the module's `_help` hook. For examples, check the story, forum or blog module.
- The `_search` hook was changed to not only return the result set array, but a two element array with the result group title and the result set array. This provides more precise control over result group titles.
- The `_head` hook is eliminated and replaced with the `drupal_set_html_head()` and `drupal_get_html_head()` functions. You can add JavaScript code or CSS to the HTML head part with the `drupal_set_html_head()` function instead.
- See also the description of the `_compose_tips` hook changes below.

Emitting links

- The functions `url()` and `l()` take a new `$fragment` parameter. Calls to `url()` or `l()` that have '#' in the `$url` parameter need to be updated. If you don't update such calls, Drupal's path aliasing won't work for URLs with # in them.
- Drupal now emits relative URLs instead of absolute URLs. Contributed modules must be updated when an absolute url is required. For example:
 - - Any module that outputs an RSS feed without using `node_feed()` should be updated. Note: this is discouraged. please use `node_feed()` instead. Also modules using `node_feed()` should provide an absolute link in the 'link' key, if any.
 - Any module which send email should be updated so that links in the email have absolute urls instead of relative urls. You do this using a parameter in your call to `l()` or `url()`

Status and error messages

- Modules that use `theme('error', ...)` to print error messages should be updated to use `drupal_set_message(..., 'error')` unless used to print an error message below a form item.

<?php

```

drupal_set_message(t('failed to update X', 'error')); // 
set the second parameter to 'error'
?>
○ Modules that print status messages directly to the screen using status()
should be updated to use drupal_set_message(). The status() function
has been removed.
<?php
drupal_set_message(t('updated X'));
?>

```

Converting 4.4 modules to 4.5

Menu system

The Drupal menu system got a complete rewrite. The new features include:

- The administrator may now customize the menu to reorder, remove, and add items.
- Menu items may be classified as "local tasks," which will by default be displayed as tabs on the page content.
- The menu API is much more consistent with the rest of Drupal's API.

The menu() function is no more. In its place, we have hook_menu(). The old hook_link() remains, but will no longer be called with the "system" argument. The hook reference in the Doxygen documentation details all the specifics of this new hook. In short, rather than making many calls to menu() in your hook_link() implementation, you will implement hook_menu() to return an array of the menu items you define.

As an example, the old pattern:

```

<?php
function blog_link($type, $node = 0, $main) {
  global $user;
  if ($type == 'system') {
    menu('node/add/blog', t('blog entry'),
user_access('maintain personal blog') ? MENU_FALLTHROUGH :
MENU_DENIED, 0);
    menu('blog', t('blogs'), user_access('access content') ?
'blog_page' : MENU_DENIED, 0, MENU_HIDE);
    menu('blog/'. $user->uid, t('my blog'), MENU_FALLTHROUGH,
1, MENU_SHOW, MENU_LOCKED);
    menu('blog/feed', t('RSS feed'), user_access('access
content') ? 'blog_feed' : MENU_DENIED, 0, MENU_HIDE,
MENU_LOCKED);
  }
}

```

```
?>
becomes:
<?php
function blog_menu($may_cache) {
    global $user;
    $items = array();
    if ($may_cache) {
        $items[] = array('path' => 'node/add/blog', 'title' =>
t('blog entry'),
            'access' => user_access('maintain personal blog'));
        $items[] = array('path' => 'blog', 'title' => t('blogs'),
            'callback' => 'blog_page',
            'access' => user_access('access content'),
            'type' => MENU_SUGGESTED_ITEM);
        $items[] = array('path' => 'blog/'. $user->uid, 'title'
=> t('my blog'),
            'access' => user_access('maintain personal blog'),
            'type' => MENU_DYNAMIC_ITEM);
        $items[] = array('path' => 'blog/feed', 'title' => t('RSS
feed'),
            'callback' => 'blog_feed',
            'access' => user_access('access content'),
            'type' => MENU_CALLBACK);
    }
    return $items;
}
?>
```

Drupal now distinguishes between 404 (Not Found) pages and 403 (Forbidden) pages. To accommodate this, modules should abandon the practice of not declaring menu items when access is denied to them. Instead, they should set the "access" attribute of their newly-declared menu item to FALSE. This will have the effect of the menu item being hidden, and also preventing the callback from being invoked by typing in the URL. Modules may also want to take advantage of the drupal_access_denied() function, which prints a 403 page (the analogue of drupal_not_found(), which prints a 404).

Path changes

Some internal URL paths have changed; check the links printed by your code. Most significant is that paths of the form "node/view/52" are now "node/52" instead, while "node/edit/52" becomes "node/52/edit".

Node changes

- The database field `static` has been renamed to `sticky`.
- Error handling of forms (such as node editing forms) is now done using `form_set_error()`. It simplifies the forms and validation code; however, it does change the node API slightly:

- The `_validate` hook and the `_nodeapi('validate')` hook of the node API no longer take an "error" parameter, and should no longer return an error array. To set an error, call `form_set_error()`.
- Node modules' `hook_form()` implementations no longer take an "error" parameter and should not worry about displaying errors. The same applies to `hook_nodeapi('form_post')` and `hook_nodeapi('form_pre')`.
- All of the `form_` family of functions can take a parameter that marks the field as required in a standard way. Use this instead of adding that information to the field description.
- In order to allow modules such as `book.module` to inject HTML elements into the view of nodes safely, `hook_nodeapi()` was extended to respond to the 'view' operation. This operation needs to be invoked after the filtering of the node, so `hook_view()` was changed slightly to no longer require a return value. Instead of calling `theme('node', $node)` and returning the result as before, the hook can just modify `$node` as it sees fit (including running `$node->body` and `$node->teaser` through the filters, as before), and the calling code will take care of sending the result to the theme. Most modules will just work under the new semantics, as the return value from the hook is just discarded, but the `$node` parameter is now required to be passed by reference (this was common but optional before).
- We have node-level access control now! This means that node modules need to make very small changes to their `hook_access()` implementations. The check for `$node->status` should be removed; the node module takes care of this check. A value should only be returned from this hook if the node module needs to override whatever access is granted by the `node_access` table. See the hook API for details.

Node listing queries need to be changed as well, so that they properly check for whether the user has access to the node before listing it. Queries of the form

```
<?php
db_query('SELECT n.nid, n.title FROM {node} n WHERE
n.status = 1 AND foo');
?>
become
<?php
db_query('SELECT n.nid, n.title FROM {node} n '.
node_access_join_sql() .' WHERE n.status = 1 AND '. 
node_access_where_sql() .' AND foo');
?>
```

See node access rights in the Doxygen reference.

Filtering changes

This change affects non-filter modules as well! Please read on even if your module does not filter.

The filter system was changed to support multiple input formats. Each input format houses an entire filter configuration: which filters to use, in what order and with what settings. The filter system now supports multiple filters per module as well.

Check_output() changes

Because of the multiple input formats, a module which implements content has to take care of managing the format with each item. If your module uses the node system and passes content through `check_output()`, then you need to do two things:

- Pass `$node->format` as the second parameter to `check_output()` whenever you use it.
- Add a filter format selector to `hook_form` using a snippet like:

```
<?php
$output .= filter_form('format', $node->format);
?>
```

The node system will automatically save/load the format value for you.

If your module provides content outside of the node system, you can decide if you want to support multiple input formats or not. If you don't, the default format will always be used. However, if your module accepts input through the browser, it is strongly advised to support input formats!

To do this, you must:

- Provide a selector for input formats on your forms, using `filter_form()`.
- Validate the chosen input format on submission, using `filter_access()`.
- Store the format ID with each content item (the format ID is a number).
- Pass the format ID to `check_output()`.

Check the API documentation for these functions for more information on how to use them.

Filter hook

The `_filter` hook was changed significantly. It's best to start with the following framework:

```
<?php
function hook_filter($op, $delta = 0, $format = -1, $text =
'') {
    switch ($op) {
```

```

        case 'list':
            return array(0 => t('Filter name'));
        case 'description':
            return t("Short description of the filter's actions.");
    /*
        case 'no cache':
            return true;
    */
        case 'prepare':
            $text = ...
            return $text;
        case 'process':
            $text = ...
            return $text;
        case 'settings':
            $output = ...;
            return $output;
        default:
            return $text;
    }
}
?>

```

When converting a module to 4.5, you can normally ignore the \$delta parameter: it is used to have multiple filters inside one module. The 'prepare', 'process' and 'settings' operations still work the same as before, with only small changes.

However, you should now include the \$format parameter in the variable names for filter settings. If your filter has a setting "myfilter_something", it should be changed to "myfilter Something \$format". This allows the setting to be set separately for each input format. To check if it works correctly, add your filter to two different input formats and give each instance different settings. Verify that each input format retains its own settings.

Unlike before, the 'settings' operation should only be used to return actually useful settings, because there is now a separate overview of all enabled filters. A filter does not need its own on/off toggle. If a filter has no configurable settings, it should return nothing for the settings, rather than a message like we did before.

Finally, the filter system now includes caching. If your filter's output is dynamic and should not be cached, uncomment the 'no cache' snippet. Only do this when absolutely necessary, because this turns off caching for any input format your filter is used in. Beware of the filter cache when developing your module: it is advised to uncomment 'no cache' while developing, but be sure to remove it again if it's not needed.

Filter tips

Filter tips are now output through the format selector. Modules no longer need to call filter_tips_short() to display them.

A module's filter tips are returned through the filter_tips hook:

```
<?php
function hook_filter_tips($delta, $format, $long = false) {
  if ($long) {
    return t("Long tip");
  }
  else {
    return t("Short tip");
  }
?>
```

As in the filter hook you can ignore the \$delta parameter if you're upgrading an existing module. If your filter's tips depend on its settings, make sure you use \$format to retrieve the setting for the current input format. \$long tells you whether to return long or short tips.

Other changes

In addition to the above mentioned changes:

- hook_user() was changed to allow multiple pages of user profile information. The new syntax of the hook is given in the API reference. Pay particular attention to the "categories", "form", and "view" operations.
- When processing a form submission, you should use drupal_goto() to redirect to the result if the submission was accepted. This prevents a double post when people refresh their browser right after submitting. Messages set with drupal_set_message() will be saved across the redirect. If a submission was rejected, you should not use drupal_goto(), but simply print out the form along with error messages.

Converting 4.5 modules to 4.6

Block system

Every block now has a configuration page to control block-specific options. Modules which have configurations for their blocks should move those into hook_block().

The only required changes to modules implementing hook_block() is to be careful about what is returned. Do not return anything if \$op is not 'list' or 'view'. Once this change is made, modules will still be compatible with Drupal 4.5.

If a specific block has configuration options, implement the additional \$op options in your module. The implementation of 'configure' should return a string containing the configuration form for the block with the appropriate \$delta. 'save' will have an additional \$edit argument, which will contain the submitted form data for saving.

Search system

The search system got a significant overhaul.

Node indexing now uses the node's processed and filtered output, which means that any custom node fields will automatically be included in the index, as long as they are visible to normal users who view the node. Modules that implement hook_search() and hook_update_index() just to have extra node fields indexed no longer need to do this.

If you wish to have additional information indexed that is *not* visible in the node display at *node/id*, then you can do so using nodeapi('update index'). If you want to add extra information to the node results, use nodeapi('search result').

However, the standard search is still limited to a keyword search. Modules that implement custom, specific search forms (like project.module) can still do so. Custom search forms that do not use hook_search() should be located/moved to a local task under the /search page.

If you are unsure of what you need to do, please refer to the complete search documentation.

Module paths

The function module_get_path was renamed to drupal_get_path which now returns the path for all themes, theme engines and modules. Because of this abstraction you must pass an additional parameter identifying the type of item for which the path is requested. The following example compares retrieving the path to image module between Drupal 4.5 and 4.6.

```
<?php
// Drupal 4.5:
$path = module_get_path('image');
// Drupal 4.6:
$path = drupal_get_path('module', 'image');
?>
```

All instances of module_get_path should be renamed to drupal_get_path.

Database backend

The function `check_query` was renamed to `db_escape_string` and now has a database specific implementation. All instances of `check_query` should be renamed to `db_escape_string`.

Theme system

The function `theme_page()` no longer takes `$title` or `$breadcrumb` arguments. Set page titles using `hook_menu()` or, if the title must be dynamically determined, use `drupal_set_title()`. Set breadcrumb trails first using `hook_menu()`, which can be overridden with `menu_set_location()` and `drupal_setBreadcrumb()`.

Watchdog messages

The `watchdog()` function now takes a severity attribute, so `watchdog($type, $message, $link)`; becomes `watchdog($type, $message, $severity, $link)`. Specify a severity in case you are reporting a warning or error. Possible severity constants are: `WATCHDOG_NOTICE`, `WATCHDOG_WARNING` and `WATCHDOG_ERROR`. Also make sure that you provide the type as a literal string, so translation extraction can pick it up.

If you are unsure of which severity to use, remember these rules:

- If the problem is caused by a definite fault and should be fixed as soon as possible, use an error message.
- If the problem could point to a fault, but could also be harmless, use a warning message. This type should also be used whenever the problem could be caused by a remote server (example: ping timeout, failed to aggregate a feed, etc).
- Normal messages should be notices.

Node markers

If you have a module calling `theme('mark')`, note that it is now possible to have different markers for different states of a node. The supported states are `MARK_NEW`, `MARK_UPDATED` and `MARK_READ`. You can get the marker state from `node_mark()`, which replaces the `node_new()` function available in previous Drupal versions.

Control over destination page after form processing

Occasionally a module might want to specify where a user should go after he submits a form. This is now possible by passing a querystring parameter `&destination=<path>`. For example, editing of nodes and comments from within the Admin pages now returns the user to those pages after he is done. For example usage, search `drupal_get_destination()` which can be found in

path.module, node.module, comment.module, and user.module

Confirmation messages

Confirmations for dangerous actions should now be presented with the theme('confirm') function for consistency. Check the function's documentation or look at some of the core modules for examples.

Note that this is a themable function which should be invoked through theme('confirm') and not theme_confirm().

Inter module calls

New features are available -- it's not necessary to use them. Now you can really (and should) use module_invoke to call a function from another module. For example, taxonomy_get_tree should be called by module_invoke('taxonomy', 'get_tree') If you need to loop through the implementations of a hook, please check the new module_implements function.

Node queries

If you have a module which retrieves a list of nodes by issuing its own database query, then the following applies.

The functions node_access_join_sql() and node_access_where_sql() should not be used any more but the SELECT-queries should be wrapped in a db_rewrite_sql() call.

If you have used DISTINCT(nid) -- because of node_access_join_sql() -- you no longer need it, replace it simply with n.nid. If you have SELECT *, please replace it with SELECT n.nid, n.* -- and always make sure that n.nid field comes first in the SELECT statement -- this way the db_rewrite_sql() function can rewrite the query to use DISTINCT(nid) should there be a need for it. If the n.nid field is not first, the query will fail when node access modules are enabled. Also, at the moment db_rewrite_sql can not handle AS -- either leave it out or lowercase it.

Always use table name before the field names, especially before nid because other tables may be JOINed during the rewrite process.

Example:

```
<?php
// Drupal 4.5:
$nodes = db_query_range('SELECT DISTINCT(n.nid) FROM {node} n
'. node_access_join_sql().' WHERE '. node_access_where_sql()
.' AND n.promote = 1 AND n.status = 1 ORDER BY n.created
DESC', 0, 15);
// Drupal 4.6:
$nodes = db_query_range(db_rewrite_sql('SELECT n.nid FROM
{n} n WHERE '. node_access_where_sql()
.' AND n.promote = 1 AND n.status = 1 ORDER BY n.created
DESC'), 0, 15);
```

```
{node} n WHERE n.promote = 1 AND n.status = 1 ORDER BY n.created DESC'), 0, 15);
?>
```

If you are not using the node table, then you shall pass the table name from which you SELECTing the nodes. For example

```
<?php
$result = db_query(db_rewrite_sql("SELECT f.nid, f.* from
{files} f WHERE filepath = '%s'", 'f'), $file);
?>
```

note the 'f' parameter of db_rewrite_sql().

Avoid USING because there could be JOINS before it, which will break the USING clause.

Text output

Drupal's text output was audited and several escaping bugs were found. For more info, see the check_plain patch.

You need to pay attention that all user-submitted plain-text in your module is escaped using check_plain() when you output it into HTML. No escaping should be done on data that is going into the database: only escape when outputting to HTML.

Check_plain() replaces drupal_specialchars() and check_form(), so if you are using any of those two, you should use check_plain() instead.

You should also wrap user-submitted text in messages with theme('placeholder', \$text). For example for "created term %term".

Pay attention in particular to node and comment titles as their behaviour has been changed. They are now stored as plain-text, like other single-line fields in Drupal and should be escaped when output. However, the function l() now takes plain-text by default instead of HTML, which means that whenever \$node->title is used as the caption for a link, it will automatically be escaped. When outputting titles literally, you still have to escape them yourself.

URLs also require attention, as the URL functions (url, request_uri, referer_uri, etc) were changed to output 'real' URLs rather than HTML-escaped URLs. When putting any of them inside an HTML tag attribute (e.g.), you need to pass it through check_url() first. When putting an URL into HTML outside of a tag or attribute, you can use check_url() or check_plain(), it doesn't matter. Don't use check_url() in situations where a real URL is expected (e.g. the HTTP "Location: ..." header).

The best test is to submit forms with HTML tags in the plain-text/single-line fields (e.g. "<u>test</u>"). If the underline tag is not interpreted, but displayed literally, your module is escaping the text correctly.

Nothing has changed for filtered/rich text, which still uses check_output() like before.

Converting 4.6 modules to HEAD

Taxonomy API change

In order to provide more meaningful messages to the user, you are now required to provide your own when using the taxonomy APIs to create or modify terms and vocabularies. This applies to taxonomy_save_vocabulary() and taxonomy_save_term().

A status message is returned, which can be either SAVED_NEW, SAVED_UPDATED or SAVED_DELETED.

This snippet shows you an example of handling this:

```
<?php
// Drupal 4.6
taxonomy_save_vocabulary($edit);
// Drupal 4.7
switch (taxonomy_save_vocabulary($edit)) {
  case SAVED_NEW:
    drupal_set_message(t('Created new vocabulary %name.', array('%name' => theme('placeholder', $edit['name']))));
    break;
  case SAVED_UPDATED:
    drupal_set_message(t('Updated vocabulary %name.', array('%name' => theme('placeholder', $edit['name']))));
    break;
  case SAVED_DELETED:
    drupal_set_message(t('Deleted vocabulary %name.', array('%name' => theme('placeholder', $deleted_name))));
    break;
}
?>
```

Table API change

Themes tables sometimes were called with arguments set to NULL or an empty string to indicate that there were either no rows or no header. This has now to be an empty array.

Change

```
<?php
theme('table', '', $rows);
?>
```

```
to
<?php
theme('table', array(), $rows);
?>
```

Join forces

Too often new modules are contributed that do nothing new, only do it in a different way. We are then stuck with two modules that offer nearly similar functionality, but both do not do it well enough. This leads to confusion, clutter and a lot of inefficiency.

So please consider the following guidelines or ideas:

- Develop and use one central API. do not introduce any new .incs, .modules or other files with APIs, if there are modules that have these already.
- Consult other developers of modules in your domain when you plan to add features, or plan to add a module. try to agree on features, to avoid overlapping. Nothing is more confusing for a user when he has, for example a Spam Queue for comments, and a completely different one for trackbacks, which does not respect the options you set for comments. Even worse, but certainly not unheard of, is that module Foo breaks module Bar, because they want to do the same, or want to use the same database tables.
- Do not try to duplicate functionality because "you do not really like how its done there" That only adds clutter. Rather improve the existing one, then introduce yet another-half-witted-module.

Note that they are not rules or laws. But that respecting them will most often help you and the community better. For only then will we be able to "stand on the shoulders of Giants" as they say in Open Source Land. If you keep reinventing wheels, you will be stuck with lots of incompatible and half finished wheels, in the end. When you use existing wheels and build a car on top of them, you will be able to get somewhere, one day.

Reference

This section is intended as a handy reference, collecting things which you may need to look up as you code to Drupal.

'Status' field values for nodes and comments

Just documenting the **status** field for the following tables

NODES

- 0: not published
- 1: published

COMMENTS

- 0: published
- 1: not published
- 2: deleted (no longer exists in Drupal 4.5 and above)

Values of 'comment' field in node table

Here are the values of the 'comment' field in the node table:

- 0 = comments cannot be added to this node and published comments will not display
- 1 = comments cannot be added to this node, but published comments will display
- 2 = new comments can be added and published comments will display

Module how-to's

This section collects various 'How-to' articles of interest to module writers and hackers.

How to write a node module

This information is superseded by the Doxygen documentation. In particular, its example node module is a good tutorial.

How to write database independent code

In order to ensure that your module works with all compatible database servers (currently Postgres and MySQL), you'll need to remember a few points.

- When you need to LIMIT your result set to certain number of records, you should use the db_query_range() function instead of db_query(). The syntax of the two functions is the same, with the addition of two required parameters at the end of db_query_range(). Those parameters are \$from and then \$count. Usually, \$from is 0 and \$count is the maximum number of records you want returned.
- If possible, provide SQL setup scripts for each supported database platform. The differences between each platform are slight - we hope documentation on these differences will be forthcoming.
- You should test any complex queries for ANSI compatibility using this tool by Mimer
- If you are developing on MySQL, use it's ANSI compatibility mode
- If you can install all database servers in your environment, it is helpful to create shell databases in each and then run sample queries in each platform's

query dispatch tool. Once your query succeeds in all tools, congratulate yourself.

- Don't use '' when you mean NULL
- Avoid table and field names that might be reserved words on any platform.
- Don't use auto-increment or SERIAL fields. Instead, use an integer field and leverage Drupal's own sequencing wrapper:
`db_next_id(<tablename_fieldname>)`

How to write efficient database JOINs

This page is based on an e-mail posted by Craig Courtney on 6/21/2003 to the drupal-devel mailing list: <http://drupal.org/node/view/322>.

There are 3 kinds of join: INNER, LEFT OUTER, and RIGHT OUTER. Each requires an ON clause to let the RDBMS know what fields to use joining the tables. For each join there are two tables: the left table and the right table. The syntax is as follows:

```
{left table} {INNER | LEFT | RIGHT} JOIN {right table} ON {join criteria}
```

An INNER JOIN returns only those rows from the left table having a matching row in the right table based on the join criteria.

A LEFT JOIN returns ALL rows from the left table even if no matching rows were found in the right table. Any values selected out of the right table will be null for those rows where no matching row is found in the right table.

A RIGHT JOIN works exactly the same as a left join but reversing the direction. So it would return all rows in the right table regardless of matching rows in the left table.

It is recommended that you **not use right joins**, as a query can always be rewritten to use left joins which tend to be more portable and easier to read.

With all of the joins, if there are multiple rows in one table that match one row in the other table, that row will get returned many times.

For example:

Table A

tid, name

1, 'Linux'

2, 'Debian'

Table B

fid, tid, message

1, 1, 'Very Cool'

2, 1, 'What an example'

Query 1:

```
SELECT a.name, b.message FROM a INNER JOIN b ON a.tid = b.tid
```

Result 1:

Linux, Very Cool

Linux, What an example

Query 2:

```
SELECT a.name, b.message FROM a LEFT JOIN b ON a.tid = b.tid
```

Result 2:

Linux, Very Cool

Linux, What an example

Debian, <null>

Hope that helps in reading some of the queries.

How to connect to multiple databases within Drupal

Drupal can connect to different databases with elegance and ease!

First define the database connections Drupal can use by editing the \$db_url string in the Drupal configuration file (settings.php for 4.6 and above, otherwise conf.php). By default only a single connection is defined

```
<?php
$db_url = 'mysql://drupal:drupal@localhost/drupal';
?>
```

To allow multiple database connections, convert \$db_url to an array.

```
<?php
$db_url['default'] =
'mysql://drupal:drupal@localhost/drupal';
$db_url['mydb'] = 'mysql://user:pwd@localhost/anotherdb';
$db_url['db3'] = 'mysql://user:pwd@localhost/yetanotherdb';
?>
```

Note that database storing your Drupal installation should be keyed as the default connection.

To query a different database, simply set it as active by referencing the key name.

```
<?php
db_set_active('mydb');
db_query('SELECT * FROM other_db');
//Switch back to the default connection when finished.
db_set_active('default');
?>
```

Make sure to always switch back to the default connection so Drupal can cleanly finish the request lifecycle and write to its system tables.

How to write themable modules

Note: this page describes Drupal's theming from the code side of things.

Drupal's theme system is very powerful. You can accommodate rather major changes in overall appearance and significant structural changes. Moreover, you control all aspects of your drupal site in terms of colors, mark-up, layout and even the position of most blocks (or boxes). You can leave blocks out, move them from right to left, up and down until it fits your needs.

At the basis of this are Drupal's theme functions. Each theme function takes a particular piece of data and outputs it as HTML. The default theme functions are all named `theme_something()` or `theme_module_something()`, thus allowing any module to add themeable parts to the default set provided by Drupal. Some of the basic theme functions include: `theme_error()` and `theme_table()` which as their name suggest return HTML code for an error message and a table respectively. Theme functions defined by modules include `theme_forum_display()` and `theme_node_list()`.

Custom themes can implement their own version of these theme functions by defining `mytheme_something()` (if the theme is named `mytheme`). For example, functions named: `mytheme_error()`, `mytheme_table()`, `mytheme_forum_display()`, `mytheme_node_list()`, etc. corresponding to the default theme functions described above.

Drupal invokes these functions indirectly using the `theme()` function. For example:

```
<?php
$node = node_load(array('nid' => $nid));
$output .= theme("node", $node);
?>
```

By default, this will call `theme_node($node)`. However, if the currently active theme is "mytheme", and this theme has defined a function `mytheme_node()`, then `mytheme_node($node)` will be invoked instead.

This simple and straight-forward approach has proven to be both flexible and fast.

However, because direct PHP theming is not ideal for everyone, we have implemented mechanisms on top of this: so-called template engines can act as intermediaries between Drupal and the template/theme. The template engine will override the `theme_functions()` and stick the appropriate content into user defined (X)HTML templates.

This way, no PHP knowledge is required and a lot of the complexity is hidden away. More information about this can be found in the Theme developer's guide,

specifically the Theming overview.

Theme developer's guide

This section of our handbook documents aspects of our theme system that will be of interest to theme developers.

Theming overview

Note: this page describes the theme system from a themer's perspective. If you are a module coder looking to make your module themable, you should read this page.

As of version 4.5, Drupal's theme system is very flexible. The new structure makes it easy to plug components together to form your theme: templating engines, templates, stylesheets and PHP.

Here's how some existing themes are built:

| Theme | Engine (PHP) | Template (XHTML) | Style (CSS) |
|-----------------------|-----------------|------------------|-------------|
| Pushbutton | XTemplate | .xtmpl | .css |
| Box Grey | | | .css |
| Box Cleanslate | PHPTemplate | .tpl.php | .css |
| Bluebeach | | .tpl.php | .css |
| Chameleon | | | .css |
| Marvin | Chameleon.theme | | .css |

A 'theme' is now an abstract thing, which can be formed in several ways:

- PHP .theme file containing overrides for theme_functions: e.g. Chameleon
- Template file (.xtmpl, .tpl.php) for a templating engine (XTemplate, PHPTemplate, ...): e.g. Pushbutton, Bluebeach
- Style sheet for an existing template or theme: e.g. Marvin, Box Cleanslate

The directory structure for the example above looks like this:

```
themes/engines/xtemplate/xtemplate.engine
themes/engines/phptemplate/phptemplate.engine
themes/pushbutton/xtemplate.tpl
themes/pushbutton/style.css
themes/box_grey/page.tpl.php
themes/box_grey/style.css
themes/box_grey/box_cleanslate/style.css
themes/bluebeach/page.tpl.php
```

```
themes/bluebeach/style.css  
themes/chameleon/chameleon.theme  
themes/chameleon/style.css  
themes/chameleon/marvin/style.css
```

Themes and templates are placed in their own subdirectory in the themes directory. The theme engines will scan every subdirectory for template files (.xtmpl, .tpl.php,...). If a style.css file is present, it will also be used.

You can also make CSS-only themes by making a subdirectory in any theme directory and placing a new style.css file in it. Drupal will combine the new stylesheet with the template it belongs in, and make it available as a new theme. This is how the Marvin and Box Cleanslate themes work.

Finally, if there is a screenshot.png file in the theme directory, Drupal will show it in the theme administration screen.

Creating custom themes

If you want to create a custom theme, you can either customize an existing theme or start from scratch.

To customize an existing theme, just copy it to a new directory in themes, and give it a unique name. Themes should not have a name that is the same as any of the default modules in Drupal or any custom modules you might have enabled or configured. Then modify the copy as much as you want. Depending on whether the theme is template or .theme-file based, you can use PHP or XHTML/CSS to modify it. As explained above, if you only want to alter the CSS of a theme, then just place a new style.css file in a subdirectory of the theme: it will appear as a new theme in Drupal.

If you want to start from scratch, there are several ways to go. If you're not a programmer, then the easiest solution is to use one of the template engines. By default, Drupal comes with the XTemplate theme engine, which requires you to create an (X)HTML skeleton with special markers. See the XTemplate documentation for more info. There are other template engines available in the contributions repository (e.g. PHPTemplate).

Drupal themes used to be coded directly in PHP. This method is still available, but is harder to use and maintain than template-based themes.

PHPTemplate theme engine

PHPTemplate is a theme engine written by Adrian Rossouw (who is also behind the theme reforms in Drupal 4.5).

It uses individual `something.tpl.php` files to theme Drupal's `theme_something()` functions. Drupal's themeable functions are documented on the Development Plumbing site. Every file contains an HTML skeleton with some simple PHP statements for the dynamic data. Thus, PHPTemplate is an excellent choice for theming if you know a bit of PHP: with some basic PHP snippets, you can create advanced themes easily.

If you don't know PHP, then PHPTemplate can still be a good choice because only small bits of code are involved. They can just be copy/pasted into your template.

An extended Forum discussion provides some of the reasoning behind the creation of PHPTemplate.

Installing PHPTemplate

The engine that runs PHPTemplate is not included in the default installation of Drupal. To use themes that use PHPTemplate (e.g. Box_grey, Kubrick, Persian) you must have the PHPTemplate engine installed. To set this up:

1. Download the latest release of the PHPTemplate Engine
2. Upload this folder into the `drupal_base/themes/engines` directory on your site.

You will now be able to use and configure PHPTemplate themes.

Creating a new PHPTemplate

To create a new PHPTemplate, create a new directory under your themes directory, for example `themes/mytheme`. Then, you need to create a file called `page.tpl.php` in that directory.

This is the only file which is absolutely required. It overrides the `theme('page')` function, which outputs the final page contents, along with all the extra decorations like a header, tabs, breadcrumbs, sidebars and a footer.

You can create files to override the following functions:

- `theme('page')` (`page.tpl.php`): theme a page
- `theme('block')` (`block.tpl.php`): theme a block in sidebar
- `theme('box')` (`box.tpl.php`): theme a generic container for the main area
- `theme('comment')` (`comment.tpl.php`): theme a comment
- `theme('node')` (`node.tpl.php`): theme a node

The PHPTemplate package contains example template files for most of these, see `box_grey` for an example of `page.tpl.php`. Simply copy them into your `theme/mytheme` directory and edit them. Note that you will need to visit `administer > themes` for PHPTemplate to refresh its cache and recognize any new `.tpl.php` files.

If you want to theme a function other than the defaults listed here, you need to provide an override yourself.

Block.tpl.php

Lays out content for blocks (left and/or right side of page). This template is optional, and can be overridden by copying the default template and modifying it.

Available variables

- \$block (object)
 - \$block->module : The name of the module that generated the block.
 - \$block->delta : The number of the block, in the module.
 - \$block->subject : The block title.
 - \$block->content : The html content for the block.
 - \$block->status : Status of block (0, or 1).
 - \$block->path : The path that matches whether or not a block is displayed.
 - \$block->region : Left (0), or Right(1) column.
 - \$block->throttle: Throttle setting.
- \$seqid : The sequential id of the block displayed, ie: The first block is 1, the second block is 2 etc.
- \$block_seqid : The same as \$seqid, but is reset for the left and right sidebars.
- \$zebra : Whether or not the block is odd , or even. This is useful for creating 'zebra stripes' with your css. This value will be either 'odd', or 'even'.
- \$block_zebra : The same as \$zebra, but is reset for the left and right sidebars.

Default template

The default `block.tpl.php`, which can be found at
themes/engines/phptemplate/block.tpl.php.

```
<div class="<?php print "block block-$block->module" ?>"  
id="<?php print "block-$block->module-$block->delta"; ?>">  
  <h2><?php print $block->subject ?></h2>  
  <div class="content"><?php print $block->content ?></div>  
</div>
```

Box.tpl.php

Prints a simple html box around a page element. For instance: The comment view options are surrounded by `box.tpl.php`.

Available variables

- \$title: The title of the box.
- \$content: The content of the box.
- \$region: Region. main, left or right.

Default template

```
<div class="box">
  <h2><?php print $title ?></h2>
  <div class="content"><?php print $content ?></div>
</div>
```

Comment.tpl.php

Define the HTML for a comment block. This doesn't have anything to do with comment threading, just the actual comment.

Available variables

- \$new : Translated text for 'new', if the comment is infact new.
- \$comment(object) : Comment object as passed to the theme_comment function.
- \$submitted : Translated post information string.
- \$title : Link to the comment title.
- \$picture : User picture HTML (include <a> tag.) , if display is enabled and picture is set.
- \$links : Contextual links below comment.
- \$content : Content of link.
- \$author : Link to author profile.
- \$date : Formatted date for post.

Default template

```
<div class="comment" <?php print ($comment->new) ?
'comment-new' : '' ?>">
<?php if ($comment->new) : ?>
  <a id="new"></a>
  <span class="new"><?php print $new ?></span>
<?php endif; ?>
<div class="title"><?php print $title ?></div>
  <?php print $picture ?>
<div class="author"><?php print $submitted ?></div>
<div class="content"><?php print $content ?></div>
  <?php if ($picture) : ?>
    <br class="clear" />
<?php endif; ?>
```

```
<div class="links"><?php print $links ?></div>
</div>
```

Node.tpl.php

This template controls the display of a node, and a node summary.

Available variables

- \$title : Title of node.
- \$node_url : Link to node.
- \$terms : HTML for taxonomy terms.
- \$name : Formatted name of author.
- \$date : Formatted data.
- \$sticky : True if the node is sticky on the front page.
- \$picture : HTML for user picture, if enabled.
- \$content : Node content, teaser if it is a summary.
- \$links : Node links.
- \$taxonomy (array) : array of taxonomy terms.
- \$node (object) : The node object.
- \$main : This variable is set to 1 if the node is being displayed on the main page, 0 otherwise.
- \$page : True if on the node view page, and not a summary.
- \$submitted : Translated text, if the node info display is enabled for this node type.

Default template

```
<div class="node"><?php print ($sticky) ? " sticky" : ""; ?>
  <?php if ($page == 0): ?>
    <h2><a href=<?php print $node_url ?>" title=<?php print
$title ?>><?php print $title ?></a></h2>
  <?php endif; ?>
  <?php print $picture ?>
  <div class="info"><?php print $submitted ?><span
class="terms"><?php print $terms ?></span></div>
  <div class="content">
    <?php print $content ?>
  </div>
  <?php if ($links): ?>
    <?php if ($picture): ?>
      <br class='clear' />
    <?php endif; ?>
    <div class="links"><?php print $links ?></div>
  <?php endif; ?>
</div>
```

Theme distinct node types differently

You can easily use PHPTemplate to produce specialized themes for specific node types. For example, to theme forum posts separately from your other nodes, use node-forum.tpl.php. This will work for any node type.

Here is the basic template.

```
node-<node type>.tpl.php
```

Page.tpl.php

This template defines the main skeleton for the page.

Available variables

- head_title: The text to be displayed in the page title.
- language: The language the site is being displayed in.
- site: The name of the site, always filled in.
- head: HTML as generated by drupal_get_html_head() (needed to dynamically add scripts to pages)
- onload_attributes: Onload tags to be added to the head tag, to allow for autoexecution of attached scripts.
- directory: The directory the theme is located in , ie: themes/box_grey or themes/box_grey/box_cleanslate
- logo: The path to the logo image, as defined in theme configuration.
- site_name: The site name of the site, to be used in the header, empty when display has been disabled.
- site_slogan: The slogan of the site, empty when display has been disabled.
- search_box: True(1) if the search box has been enabled.
- search_url: URL the search form is submitted to.
- search_button_text: Translated text on the search button.
- search_description: Translated description for the search button.
- title: Title, different from head_title, as this is just the node title most of the time.
- primary_links (array): An array containing the links as they have been defined in the phptemplate specific configuration block.
- secondary_links (array): An array containing the links as they have been defined in the phptemplate specific configuration block.
- breadcrumb: HTML for displaying the breadcrumbs at the top of the page.
- tabs: HTML for displaying tabs at the top of the page.
- messages: HTML for status and error messages, to be displayed at the top of the page.
- layout: This setting allows you to style different types of layout ('none', 'left', 'right' or 'both') differently, depending on how many sidebars are enabled.
- help: Dynamic help text, mostly for admin pages.

- styles: Required for stylesheet switching to work. This prints out the style tags required.
- mission: The text of the site mission.
- is_front: True if the front page is currently being displayed. Used to toggle the mission.
- sidebar_left: The HTML for the left sidebar.
- content: The HTML content generated by Drupal to be displayed.
- sidebar_right: The HTML for the right sidebar.
- footer_message: The footer message as defined in the admin settings.
- closure: Needs to be displayed at the bottom of the page, for any dynamic javascript that needs to be called once the page has already been displayed.

Default template

Here is the contents of the box_grey template's page.tpl.php, to give you an idea of the layout of the file.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title><?php print $title ?></title>
  <meta http-equiv="Content-Type" content="text/css" />
  <?php print $head ?>
  <?php print $styles ?>
</head>
<body <?php print theme("onload_attribute"); ?>>
<div id="header">
  <?php if ($search_box): ?>
  <form action=<?php print url("search") ?>" method="post">
    <div id="search">
      <input class="form-text" type="text" size="15" value="" name="keys" /><input class="form-submit" type="submit" value=<?php print t("Search") ?>" />
    </div>
  </form>
  <?php endif; ?>
  <?php if ($logo) : ?>
    <a href=<?php print url() ?>" title="Index Page"><img src=<?php print($logo) ?>" alt="Logo" /></a>
  <?php endif; ?>
  <?php if ($site_name) : ?>
    <h1 id="site-name"><a href=<?php print url() ?>" title="Index Page"><?php print($site_name) ?></a></h1>
  <?php endif; ?>
  <?php if ($site_slogan) : ?>
```

```

        <span id="site-slogan"><?php print($site_slogan)
?></span>
<?php endif;?>
<br class="clear" />
</div>
<div id="top-nav">
<?php if (is_array($secondary_links)) : ?>
<ul id="secondary">
<?php foreach ($secondary_links as $link): ?>
<li><?php print $link?></li>
<?php endforeach; ?>
</ul>
<?php endif; ?>
<?php if (is_array($primary_links)) : ?>
<ul id="primary">
<?php foreach ($primary_links as $link): ?>
<li><?php print $link?></li>
<?php endforeach; ?>
</ul>
<?php endif; ?>
</div>
<table id="content">
<tr>
<?php if ($sidebar_left != ""): ?>
<td class="sidebar" id="sidebar-left">
<?php print $sidebar_left ?>
</td>
<?php endif; ?>
<td class="main-content" id="content-<?php
$layout ?>">
<?php if ($title != ""): ?>
<h2 class="content-title"><?php print $title
?></h2>
<?php endif; ?>
<?php if ($tabs != ""): ?>
<?php print $tabs ?>
<?php endif; ?>
<?php if ($mission != ""): ?>
<p id="mission"><?php print $mission ?></p>
<?php endif; ?>
<?php if ($help != ""): ?>
<p id="help"><?php print $help ?></p>
<?php endif; ?>
<?php if ($messages != ""): ?>
<div id="message"><?php print $messages ?></div>
<?php endif; ?>
<!-- start main content -->
```

```

<?php print($content) ?>
<!-- end main content -->
</td><!-- mainContent -->
<?php if ($sidebar_right != ""): ?>
<td class="sidebar" id="sidebar-right">
    <?php print $sidebar_right ?>
</td>
<?php endif; ?>
</tr>
</table>
<?php if ($breadcrumb != ""): ?>
    <?php print $breadcrumb ?>
<?php endif; ?>
<div id="footer">
    <?php if ($footer_message) : ?>
        <p><?php print $footer_message;?></p>
    <?php endif; ?>
Validate <a
href="http://validator.w3.org/check/referer">XHTML</a> or <a
href="http://jigsaw.w3.org/css-validator/check/referer">CSS</a>.
</div><!-- footer -->
    <?php print $closure;?>
</body>
</html>

```

Alternative templates for different node types

There are times when you may want to create a static page within Drupal such as an "about" page, help pages and the like. Obviously for these, you don't want the title, author links, or indeed anything except the page content.

To accomplish this, copy your node.tpl.php to node-\$type.tpl.php. In the case of a static page, you would copy your node.tpl.php file to node-page.tpl.php.

The path module will then allow you to type in "clean URLs" like "about", "bio", etc.

This feature was added to the last release of the 4.5 release of PHPTemplate, so make sure that your phptemplate is current if this doesn't work for you.

Example - Theming flexinode

Since it took me a while to make sense of this I thought I would post an example to help others along the way. It's actually quite simple, just not very intuitive.

This example is for changing the way that the flexinode 'date/time' field will display on a page. (I only wanted month and year to show). But could very easily be adapted to other things.

My theme is called 'licc' - it is a phptemplate theme.

The Quick Version

In the directory for your phptemplate theme (this could be an existing or custom theme), create the following 2 files. *For me the files were put in 'themes/licc'.*

Create template.php

Something like this:

```
<?php
/**
 * Override theme_flexinode_timestamp() from
 modules/flexinode/field_timestamp.inc
 */
function phptemplate_flexinode_timestamp($field_id, $label,
 $value, $formatted_value) {
 // nothing happens here.
 return _phptemplate_callback('flexinode_timestamp',
 array('field_id' => $field_id, 'label' => $label, 'value' =>
 $value, 'formatted_value' => $formatted_value));
}
?>
```

Create flexinode_timestamp.tpl.php

Something like this:

```
<?php
$formatted_value = strftime ("%B %Y", $value); // format as
Month and Year, eg. 'July 2004'
?>
<div class="flexinode-timestamp-<?php print $field_id; ?>">
<strong><?php print $label; ?>: </strong><br />
<?php print $formatted_value; ?>
</div>
```

That's it! Just modify the second file so that the field is displayed the way you would like.

Note: that you will need to visit *administer > themes* for PHPTemplate to refresh its cache and recognize any new .tpl.php files.

Below is the long-winded version, read on if you are interested...

The Long Version

1. find the theme function for the flexinode field

Found in modules/flexinode/field_timestamp.inc

```
<?php
function theme_flexinode_timestamp($field_id, $label, $value,
$formatted_value) {
    $output = theme('form_element', $label, $formatted_value);
    $output = '<div class="flexinode-timestamp-' . $field_id
. '">' . $output . '</div>';
    return $output;
}
?>
```

This is just for reference, you could just as easily look in the API documentation. Core documentation is here:

<http://drupaldocs.org/api/head/group/themeable>

(only core modules seem to be online at the moment, so you will need to search through the code for any add-on modules like flexinode)

If you wanted to theme flexinode 'image' fields, you would need to look for the theme function in modules/flexinode/field_image.inc

2. Create template.php and add override function

For my theme I created themes/licc/template.php and then copied the function declaration from above replacing the word 'theme' with 'phptemplate'.

```
<?php
function phptemplate_flexinode_timestamp($field_id, $label,
$value, $formatted_value) {
}
?>
```

add in the phptemplate callback:

```
<?php
return _phptemplate_callback('flexinode_timestamp',
array('field_id' => $field_id, 'label' =>
$label, 'value' => $value, 'formatted_value' =>
$formatted_value));
```

```
?>
```

This function doesn't really `_do_` anything except give phptemplate control over the display of this field, the next step looks after the actual formatting. Note how the variables are passed on to the `_phptemplate_callback()` in an associative array.

Note: do not use the key 'file' in the callback array, as it causes problems for phptemplate. This is a value used for the image field in particular. This is what I did (for the image field) to get around this problem (see 'imgfile' used instead of 'file')

```
<?php
/**
 * Override theme_flexinode_image() from
modules/flexinode/field_image.inc
*/
function phptemplate_flexinode_image($field_id, $label,
$file, $formatted_value) {
    // empty 'stub' function
    return _phptemplate_callback('flexinode_image',
array('field_id' => $field_id, 'label' => $la
bel, 'imgfile' => $file, 'formatted_value' =>
$formatted_value));
}
?>
```

3. Create flexinode_timestamp.tpl.php to do formatting

This goes in your theme directory (for me themes/licc/flexinode_timestamp.tpl.php). As you can see the name matches the bit after 'phptemplate_' in the theme override function, and the first argument of the `_phptemplate_callback()`.

Put the HTML/PHP that you want in this file for the display of all date/time (timestamp) fields in all flexinode pages.

Something like this:

```
<div class="flexinode-timestamp-<?php print $field_id; ?>">
<strong><?php print $label; ?>: </strong><br />
<?php print $formatted_value; ?>
</div>
```

Example Files

template.php

```
<?php
/**
 * template.php
 *
 * This file contains functions for over-riding the default
 * theme functions
 * in Drupal core and modules (look at the API documentation
 * for more info).
 * The functions don't actually _do_ anything, except pass the
 * variables
 * available to phptemplate for use in the *.tpl.php files.
 *
 * Add similar 'stub' functions to override other default
 * theme functions.
 */
/**
 * Override theme_flexinode_timestamp() from
 * modules/flexinode/field_timestamp.inc
 */
function phptemplate_flexinode_timestamp($field_id, $label,
$value, $formatted_value) {
    // like I said, nothing happens here.
    return _phptemplate_callback('flexinode_timestamp',
array('field_id' => $field_id, 'label' => $label, 'value' =>
$value, 'formatted_value' => $formatted_value));
}
?>
```

flexinode_timestamp.tpl.php

```
<?php
/**
 * Customised formatting of flexinode timestamp data in nodes.
 * These fields are available:
 * $field_id, $label, $value, $formatted_value
 */
// Change the default $formatted_value so that it suits me
// (no time or day)
$formatted_value = strftime ("%B %Y", $value); // format as
Month and Year, eg. 'July 2004'
?>
<div class="flexinode-timestamp-<?php print $field_id; ?>">
<strong><?php print $label; ?>: </strong><br />
<?php print $formatted_value; ?>
</div>
```

Making additional variables available to your templates

Examples from this forum discussion. The \$hook refers to the area the variable is to be used in (e.g. for comment.tpl.php, it would be "comment").

This function needs to be defined in a template.php file, which is placed inside the template directory (for instance :

themes/box_cleanslate/template.php)

Note: For these changes to take effect, you need to load the admin/themes page first.

```
<?php
function _phptemplate_variables($hook, $vars) {
    switch($hook) {
        case 'comment' :
            $vars['newvar'] = 'new variable';
            $vars['title'] = 'new title';
            break;
    }
    return $vars;
}
?>
```

The output of this function is merged with the variables returned from phptemplate_comment, so you can easily adjust whichever variables you feel necessary.

Your comment.tpl.php file will now have a new variable available in it called \$newvar. Similarly the \$title variable will be overridden with the value specified in the function.

A neat trick is to count how many times each of the hooks is called, so you can pass an extra variable. re :

```
<?php
function _phptemplate_variables($hook, $vars) {
    static $count;
    $count = is_array($count) ? $count : array();
    $count[$hook] = is_int($count[$hook]) ? $count[$hook] : 1;
    $vars['zebra'] = ($count[$hook] % 2) ? 'odd' : 'odd';
    $vars['seqid'] = $count[$hook]++;
    return $vars;
}
?>
```

That is 'even' if it is an even number, and 'odd' if it is odd. This means you do zebra striping (ie: alternating colors) for each of your nodes / blocks / comments / whatever.

Then you can set up a some styles for class='zebra', which handle the alternating colors.

Another example is a flag to show us if we are looking at a node. Might be handy for rendering items different, when someone is looking at an article.

```
<?php
function _phptemplate_variables($hook, $vars) {
    switch ($hook) {
        case 'page':
            if (arg(0) == 'node' && is_numeric(arg(1)) && arg(2) ==
 '') {
                $vars['content_is_node'] = TRUE;
            }
            break;
    }
    return $vars;
}
?>
```

Note that the switch is kind of obsolete here, but i leave it here, because you might want to add more variables. In that case you need them.

The args() checks will see if you have an url like /node/NID/ and not like /node/NID/edit or /node. If that is found, we set the flag TRUE.

Overriding other theme functions

If you want to override a theme function not included in the basic list (block, box, comment, node, page), you need to tell PHPTemplate about it.

To do this, you need to create a `template.php` file in your theme's directory. This file should contain the required `<?php ?>` tags, along with *stubs* for the theme overrides. These stubs instruct the engine what template file to use and which variables to pass to it.

First, you need to locate the appropriate theme function to override. You can find a list of these in the API documentation. We will use `theme_item_list()` as an example.

The function definition for `theme_item_list()` looks like this:

```
<?php
function theme_item_list($items = array(), $title = NULL) {
?>
```

Now you need to place a stub in your theme's template.php, like this:

```
<?php
/**
 * Catch the theme_item_list function, and redirect through
the template api
*/
function phptemplate_item_list($items = array(), $title =
NULL) {
    // Pass to phptemplate, including translating the
parameters to an associative array. The element names are the names
that the variables
    // will be assigned within your template.
    return _phptemplate_callback('item_list', array('items' =>
$items, 'title' => $title));
}
?>
```

We replaced the word `theme` in the function name with `phptemplate` and used a call to `_phptemplate_callback()` to pass the parameters (`$items` and `$title`) to PHPTemplate.

Now, you can create a `item_list.tpl.php` file in your theme's directory, which will be used to theme item lists. This function should follow the same logic as the original `theme_item_list()`.

Note that you will need to visit `admininster > themes` for PHPTemplate to refresh its cache and recognize the new file. Beginning with version 4.6, this is not necessary anymore.

Example - Overriding the user profile pages using PHPTemplate

This description Illustrates how easy it is to override theme functions. I'm very very new to php and sql, but even I managed to work out how to customize how user profile pages appear by overriding the theme function.

Before

This is how the out-of-the-box user profile looks like, with extra profile fields, such as City, Country, Postcode, Position etc. added in. (please note that i couldn't fit the whole page into the one screenshot..there is an extra "background/more info." field that doesn't show in the BEFORE screen shot.

[click to view the BEFORE screenshot in a new window](#)

After

This is how the exact same user profile looks after overriding the theme and applying a simple `user_profile.tpl.php` file in my theme directory.

[click to view the AFTER screenshot in a new window](#)

More details & discussion on this is in the [original forum post](#).

Not including drupal.css

You can override the `theme_stylesheet_import` function and omit `drupal.css`. Simply add this to your theme's `template.php` file:

```
<?php
function phptemplate_stylesheet_import($stylesheet, $media =
'all') {
  if ($stylesheet != 'misc/drupal.css') {
    return theme_stylesheet_import($stylesheet, $media);
  }
}
?>
```

of course, you may include your own version of `drupal.css` here but that's the basic idea.

Protecting content from anonymous users when using overrides

This is a useful tip, especially for designers or newbies to php who want to unleash the power of drupal & the power of CSS, layouts while protecting content intended for Logged In users only.

Using PHPTemplate Overrides with protected content

PHPTemplate is superb, in my opinion. I particularly like the ability to override specific layouts, but, when you override the theme function you also override/bypass permission settings -- it doesn't pass through any user access layer in Drupal. **Which is important to know if your drupal site has content that is intended for logged in users only.**

Example

In the example below, I wanted to override the way a User Profile is displayed.

Access to view User Profiles was set under ADMINISTER -> USERS -> CONFIGURE -> PERMISSIONS so that only logged in Users could view a user profile.

After the page layout override was implemented, anyone could see profile pages by guessing a link like ?=user/989 for example.

Solution

To get around that problem (and after a lot of playing around) I came up with the following solution, i.e. check to see if the user is logged in BEFORE invoking your phptemplate override. It's remarkably simple, now that I have worked it out, but, I thought it would be worth sharing on here as there maybe other drupal site administrators like me who are as thick as a plank of wood and do not have a lot of experiece in PHP programming.

I have pasted example code below that goes in the template.php file in the themes folder which invokes an override and loads a custom user_profile.tpl.php which overrides the way a user profile is displayed.

```
<?php
/**
 * check if the user is logged in before invoking the template
 * override
 */
global $user;
if($user->uid) // check to see if the user is logged in
{
  function phptemplate_user_profile($user, $fields = array()) {
    // Pass to phptemplate, including translating the
    parameters to an associative array. The element names are the names
    that the variables
    // will be assigned within your template.
    /* potential need for other code to extract field info */
    return _phptemplate_callback('user_profile', array('user' =>
    $user, 'fields' => $fields));
  }
}
?>
```

Themeing front page and others

One weakness of many Drupal sites is the *sameness* of the pages. The sections.module allows the admin to assign different themes to different areas of the site, but for a site with custom themes this requires duplication of a lot of resources such as css and image files into separate theme directories.

FactoryJoe recently turned me onto a great trick for getting much of the sections.module functionality into a single phptemplate theme.

Probably the easiest and most common use is to give the home (a.k.a. front) page an entirely different layout. It turns out that this is really easy to do. Here's how:

1. Create your home page theme template using the techniques described elsewhere in this manual. Call this file "home.tpl.php" and place it into the directory for the theme you're developing.
2. Create your second-level page template and call it "page.tpl.php". But, here's the trick, at the very top of the file, before the doctype declaration or anything, add this code:

```
<?php
if ($is_front) {
  include('home.tpl.php');
  return;
}
?>
```

This checks phptemplate's \$is_front variable and if it is true, the home.tpl.php file is imported and the 'return' line keeps the rest of the page.tpl.php code from executing.

This technique can be extended to other areas of the site by substituting other tests for \$is_front. For instance you could retheme the entire administration area, by using the following:

```
<?php
if ($is_front) {
  include('home.tpl.php');
  return;
}
elseif (arg(0)=="admin") {
  include('admin.tpl.php');
  return;
}
?>
```

Find more information about Drupal's arg() function [here](#).

Note that there shouldn't be a line break before the DOCTYPE declaration, so you should immediately follow both examples with the doctype line. Example:

```
<?php
// code //
return;
} ?><!DOCTYPE HTML PUBLIC "-//W3C//DTD HT... etc...
```

XTemplate to PHPTemplate conversion

Firstly rename the xtemplate.xtpl file to original.xtpl so that the theme is no longer a xtemplate theme.

Creating page.tpl.php

1. copy original.xtpl to page.tpl.php
2. Remove the node, comment, box, and block sections. In the place of these sections add the following code
`<?php echo $content ?>`
3. Change all the "{" characters to "<?php print \$"
4. Change all the "}" characters to "; ?>"
5. Change the "\$footer" to "\$closure"
6. Change "\$message" to "\$messages"
7. As the primary and secondary links in phptemplate are arrays you will need to change them from "echo \$primary_links;" to "echo theme('links', \$primary_links);". Also the same needs to be done for secondary links.
8. In the blocks section we need to change the \$block to either \$sidebar_left or \$sidebar_right depending on which side of the content it is on.

Creating node.tpl.php

1. Copy the node section from the original.xtpl to a new file node.tpl.php
2. As before change all the "{" and "}" characters to "<?php print \$" and "; ?>" respectively.
3. change \$link to \$node_url.
4. Change "\$taxonomy" to "\$terms"
5. Change "print \$sticky;" to "if (\$sticky) { print " sticky"; }"
6. Change "print \$picture;" to "if (\$picture) { print \$picture; }"

Creating comment.tpl.php

1. Copy the comment section from the original.xtpl to a new file comment.tpl.php
2. As before change all the "{" and "}" characters to "<?php print \$" and "; ?>" respectively.
3. Change "print \$picture;" to "if (\$picture) { print \$picture; }"

Also just to be clean, you may want to change the displaying of the new so it will only show when the \$new != "

Create block.tpl.php

1. Copy the block section from the original.xtpl to a new file comment.tpl.php
2. As before change all the "{" and "}" characters to "<?php print \$block->" and "; ?>" respectively.
3. Then change the \$block->title to \$block->subject.

Create box.tpl.php

1. Copy the box section from the original.xtpl to a new file comment.tpl.php
2. As before change all the "{" and "}" characters to "<?php print \$" and "; ?>" respectively.

XTemplate theme engine

The XTemplate theme system uses templates to layout and style Web pages. It separates **logic** (PHP), **structure** (XHTML/HTML), and **style** (CSS), making it easy for designers to create or modify templates by working on XHTML/HTML and CSS without having to worry about any PHP coding.

XTemplate templates are directories, which contain all the XHTML/HTML, CSS, image and JavaScript files that a template uses. Templates are located in the themes directory of a Drupal installation:

```
/themes/
```

Once a template exists in the themes directory, XTemplate auto-detects it, and makes it available for selection to administrators:

```
administer -> themes
```

Drupal is distributed with two XTemplate templates included - **Bluemarine** and **Pushbutton**.

Although XTemplate is still supported as part of the core, it may not be in the future, for several reasons. This will not necessarily mean the end of XTemplate since it may be maintained as an alternative contributed engine like PHPTemplate.

Creating a new XTemplate

To make a new XTemplate template, create a directory in your Drupal installation at this location:

```
/themes/
```

Whatever you name the new directory will be used as the name of your new template, for instance:

```
/themes/rembrant
```

Once you create a template in this directory, it will appear on the theme selection page as the "rembrant" template.

The easiest way to create a new template is to make a copy of an existing template, such as Default or Pushbutton, and start making changes to the files.

The only file required in a template directory is **xtemplate.xtpl**, which is a regular HTML or XHTML file containing some XTemplate tags that Drupal substitutes with content when a page is served. The xtemplate.xtpl file can be edited in DreamWeaver, GoLive, BBEdit or any other application you use to work on HTML/XHTML.

All other files in the template are optional, and are linked to from the xtemplate.xtpl file. These can include CSS, image or JavaScript files, and should all be included in the template directory to make the template easy to maintain and portable between Drupal installations.

Note that if you name your stylesheet `style.css`, it will automatically be picked up by Drupal, and you will not need to add an explicit `@import` or `<link />` for it. If you make a subdirectory within your template, containing another `style.css` file, then the subdirectory becomes a new theme, using the XHTML from the first template, but with a different stylesheet.

Template basics

xTemplate creates Web pages by substituting place holder tags in a template, the xtemplate.xtpl file, with content from the database.

There are two kinds of template place holder tags, section tags and item tags.

Section Tags

Section tags deal with the structure of a Web page, marking areas of the page, and are XHTML/HTML comment tags which look like this:

```
<!-- BEGIN: title -->
<!-- END: title -->
```

Some section tags mark areas were the content, and it's structure, will be repeated. For instance the comment section may be repeated more than once depending on how many comments are on a page:

```
<!-- BEGIN: comment -->
<!-- END: comment -->
```

Section tags can be nested, so that one set of section tags can be contained by another:

```
<!-- BEGIN: node -->
  <!-- BEGIN: title -->
    <!-- END: title -->
  <!-- END: node -->
```

Item Tags

Item tags are place holders for content items, such as the title of a page, who the page was submitted by, or the main content of a page. Item tags look like this:

```
{title}  
  
{submitted}  
  
{content}
```

Item tags are associated with the section tag that surrounds them, for instance:

```
<!-- BEGIN: node -->  
{title}  
<!-- END: node -->
```

The {title} tag above is the main title of a page, while the {title} tag below is the title for the comments on a page.

```
<!-- BEGIN: comment -->  
{title}  
<!-- END: comment -->
```

Header section

The Section

The xTemplate Header section starts and ends with these tags

```
<!-- BEGIN: header -->  
  
<!-- END: header -->
```

Don't confuse the Header section with the XHTML/HTML `<head>` element. Although the `<head>` element is included in the Header section, it also holds the top part of the Web page - the area designers usually refer to as the "Header", which usually consists of a horizontal bar with the site's logo and some navigation links.

Prolog

The WC3 recommends that all XHTML documents should start with an XML prolog specifying the encoding of the document, for instance:

```
<?xml version="1.0" encoding="utf-8"?>
```

Unfortunately there are many browsers that handle the XML prolog badly, and either crash, fail to display the page, or display it incorrectly. It is therefore recommended to leave out the XML prolog, and specify encoding in a Content-Type element in the `<head>` of your template (which Drupal does automatically).

DOCTYPE

The DOCTYPE element tells a browser two things, which XML language the document is using, and where the DTD (Document Type Declaration) of that language is located.

This is an example of a DOCTYPE element:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

There should be absolutely nothing in your document before the DOCTYPE or XML prolog. The xTemplate tag <!-- BEGIN: header --> is OK, as it will be removed by Drupal before sending the page to the browser, but make sure to **remove spaces or line breaks** between this and the DOCTYPE or XML prolog elements, or you may get unexpected results in some browsers.

To learn more about the DOCTYPE element, and which version would suit your needs best, read:

Fix Your Site With the Right DOCTYPE!
by Jeffrey Zeldman

{head_title}

Content of the <title> element. Used as the window title by browsers, and as the page title in search engine listings.

{head}

Filled in with the following:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<base href="http://yoursite.com/" />
<style type="text/css" media="all">
@import url(misc/drupal.css);
</style>
```

{styles}

Declarations for the current style:

```
<style type="text/css" media="all">@import "themes/bluemarine/style.css";</style>
```

Add this tag to allow your template to take advantage of the Drupal theme system's style-switching ability. Note that, if you have a default stylesheet, it should be named `style.css` and be located in the same directory as your `xtemplate.xtpl` file.

{onload_attributes}

The page attributes for the <body> tag.

{logo}

The logo section begins and ends with these tags:

```
<!-- BEGIN: logo -->  
<!-- END: logo -->
```

The filename for the site logo, configurable by the Administrator in the text box in the Drupal theme administration section. (Display of this item is optional.)

{site_name}

The site name section begins and ends with these tags:

```
<!-- BEGIN: site_name -->  
<!-- END: site_name -->
```

The current site name, configured by the Administrator in the text box "Name" on Drupal page:

administer->settings

(Display of this item is optional.)

{site_slogan}

The site slogan section begins and ends with these tags:

```
<!-- BEGIN: site_slogan -->  
<!-- END: site_slogan -->
```

The current site slogan, configured by the Administrator in the text box "Slogan" on Drupal page:

administer->settings

(Display of this item is optional.)

{secondary_links} {primary_links}

These tags hold whatever the Administrator inputs into the text boxes "Secondary links:" and "Primary links" in the Drupal theme administration section. If the Administrator does not specify any "Primary links", Drupal will automatically generate a set of links based on the currently-enabled modules.

The Administrator could use these tags to input links to the main sections of the site, the title of the site, a site message, an image or anything else they require.

Search Box

The Search Box section begins and ends with these tags:

```
<!-- BEGIN: search_box -->  
<!-- END: search_box -->
```

{search_url}

The form action: "search"

{search_description}

The alt text description of the search text box: "Enter the terms you wish to search for."

{search_button_text}

The value of the search submit button: "Search"

Mission

The Mission section begins and ends with these tags:

```
<!-- BEGIN: mission -->  
<!-- END: mission -->
```

{mission}

The text of the site mission statement, appears only on the Home Page, and is configured by the Administrator in the text box "Mission" on Drupal page:

```
administer->settings
```

Title

The Title section begins and ends with these tags:

```
<!-- BEGIN: title -->  
<!-- END: title -->
```

{title}

The title of the node

Tabs

The Tabs section begins and ends with these tags:

```
<!-- BEGIN: tabs -->  
<!-- END: tabs -->
```

{tabs}

Draws the Drupal "local tasks" for the current page.

{breadcrumb}

The breadcrumb trail of the page, the path from Home Page to the current page.

Help

The Help section begins and ends with these tags:

```
<!-- BEGIN: help -->  
<!-- END: help -->
```

{help}

Contains any help information which exists for a particular page.

Message

The Message section begins and ends with these tags:

```
<!-- BEGIN: message -->  
<!-- END: message -->
```

Message appears when Drupal confirms the results of an action by the user, for instance after updating or deleting a page.

{message}

The text of the message.

Node section

The Node Section

The node section (xtemplate.xtpl) contains the main content of the page, and begins and ends with these tags:

```
<!-- BEGIN: node -->
<!-- END: node -->
```

{sticky}

Sets the class to "node sticky" if a node is "stickied" at the top of lists. (i.e. if a teaser for the page is always to be displayed on the home page) If the node has not been set to be sticky, the class is set to "node".

Picture

Picture contains an image representing the user who posted the content of a node, the image is linked to the poster's profile. This is also sometimes called an "avatar". Picture begins and ends with these tags:

```
<!-- BEGIN: picture -->
<!-- END: picture -->
```

{picture}

Outputs the following:

```
<a href="user/1" title="View user profile.">
</a>
```

Title

The title of the main content of the page (node), tags begin and end:

```
<!-- BEGIN: title -->
<!-- END: title -->
```

On a node page, the title is output as:

```
<h1 class="title">Node Title</h1>
```

On the Home Page, each node title is output as:

```
<h2 class="title"><a href="node/31">Node Title</a></h2>
```

{link}

Outputs the link to the node , "node/31" in the example above.

{title}

Outputs the text of the node title, "Node Title" in the example above.

{submitted}

The username of the person who submitted the node content, outputs:

```
Submitted by <a href="user/1" title="View user profile.">Username</a> on 16 February, 2004 - 23:46.
```

Taxonomy

A list of links to taxonomies which the node belongs to, tags begin and end:

```
<!-- BEGIN: taxonomy -->  
<!-- END: taxonomy -->
```

{taxonomy}

Outputs a taxonomy term that the node belongs to:

```
<a href="taxonomy/term/30">Taxonomy Term</a>
```

{content}

The main content of the node.

Links

The control options for the node: "printer-friendly version", "add new comment", and the visitor history of the node. Tags begin and end:

```
<!-- BEGIN: links -->  
<!-- END: links -->
```

{links}

Outputs the following (depending on the viewer's permissions):

```
<a href="book/print/8"
title="Show a printer-friendly version of this book page
and its sub-pages.">printer-friendly version</a> |
<a href="comment/reply/8#comment"
title="Share your thoughts and opinions related to this posting."
>add new comment</a> |
<a href="admin/statistics/log/node/8">662 reads</a>
```

Comment

The Comment Section

The comment section (xtemplate.xtpl) contains all the comments associated with a node, and begins and ends with these tags:

```
<!-- BEGIN: comment -->
<!-- END: comment -->
```

The content of this section creates the code for a single comment, and is automatically repeated for as many times as there are comments.

Avatar

Avatar contains an image representing the user who posted the content of a node, the image is linked to the poster's profile. Avatar begins and ends with these tags:

```
<!-- BEGIN: avatar -->
<!-- END: avatar -->
```

{avatar}

Outputs the following:

```
<div class="avatar">
<a href="user/1" title="View user profile.">

</a>
</div>
```

Title

The title of a comment. Tags begin and end:

```
<!-- BEGIN: title -->
```

```
<!-- END: title -->
```

{link}

If required, changes the comment title into a link to the comment. Used when displaying comments in certain views.

{title}

The text of the comment title.

Submitted

{submitted}

Displays the username of the comment poster, linked to their profile, and the date and time the comment was posted. This is the output:

Submitted by username on Mon, 04/19/2008 - 11:56.

New

Indicates if a comment is new. Tags begin and end:

```
<!-- BEGIN: new -->  
<!-- END: new -->
```

{new}

Adds the word "new" to a comment.

Content

Displays the content of a comment.

{content}

The comment text.

Links

Displays control links for comment, such as "reply", "delete", and "edit". Tags begin and end:

```
<!-- BEGIN: links -->
<!-- END: links -->
```

{links}

Displays the control links.

Blocks

The Section

The blocks section contains the column of boxes which can be used to display various navigation and feature options, such as **Forum Topics**, **Blogs**, **Who's Online**, and **Syndicate**. Blocks sections can be configured to appear on the left or right of a page, or on both sides. The section begins and ends with this code:

```
<!-- BEGIN: blocks -->
<!-- END: blocks -->
```

{blocks}

This tag is replaced by whatever blocks have been switched on in the Administration page (admin/system/block).

Block

The block section defines the structure of each block, note the 's' in block/blocks.

```
<!-- BEGIN: block -->
<!-- END: block -->
```

{module}

The name of the module who's block is being displayed, this is added to a CSS class and ID which can be used customise the look of the block.

{delta}

Adds a number to the ID of a block, so that each block has a unique ID even if a module displays more than one block.

{title}

The title of the block.

{content}

The content of a block.

Footer

The Footer Section

The footer section appears at the very bottom of each page, it's content can be specified by the Administrator (admin/settings). The section begins and ends with this code:

```
<!-- BEGIN: footer -->
<!-- END: footer -->
```

Message

This area holds the mark-up around the message posted by the Administrator. The section begins and ends with this code:

```
<!-- BEGIN: message -->
<!-- END: message -->
```

{footer_message}

Displays the actual content defined through the field "Footer message" in the "Settings" Administration page (admin/settings).

{footer}

Outputs footer messages generated by Drupal modules. (i.e. performance statistics from devel.module)

Editing with GoLive

Set Up

To edit xTemplate template files (xtemplate.xtpl) in Adobe GoLive, follow these simple steps:

1. In the GoLive menu select "GoLive" then "Web Settings"
2. The Web Settings window will appear, click on the "File Mappings" tag.
3. In the File Mappings window open the "text/" directory
4. Scroll down until you see "html" in the Suffix column.
5. Click on "html" to select it, then click on the "+" button to create a duplicate.
6. Change the suffix of the duplicate html to "xtmpl"
7. That's it you're done!

Editing

If when opening a template file GoLive asks you which encoding to use, select "UTF-8".

If all you see after opening a template is "body onload-attributes", go into source mode and delete "{onload_attributes}" from:

```
<body{onload_attributes}>
```

Remember to add "{onload-attributes}" back once you are finished editing.

In xtemplate.xtpl, you may wish to add the following line temporarily:

```
<link type="text/css" rel="stylesheet" href="style.css" />
```

Remember to remove this line when completing work on the template, however. If you do not, Drupal will not be able to switch between various styles for your theme. Drupal will automatically load your style.css, if one exists, in the {styles} tag.

Plain PHP themes

PHP themes are the most direct way of themeing Drupal. A PHP theme consists of overrides for Drupal's built-in theme functions. You will most likely only override the basic theme hooks (pages, nodes, blocks, ...), but you can theme anything from lists to links if you desire.

To create a PHP theme, create a directory in your themes directory (we will assume `themes/mytheme` in this document), and inside that directory create a `mytheme.theme` file. This file is a regular PHP file, so make sure it contains `<?php ?>` tags.

The default theme functions in Drupal are all named `theme_something()` or `theme_module_something()`, thus allowing any module to add themeable parts to the default set provided by Drupal. Some of the basic theme functions include: `theme_error()` and `theme_table()` which as their name suggests, return HTML code for an error message and a table respectively. Theme functions defined by modules include `theme_forum_display()` and `theme_node_list()`.

In your `.theme` file, you can override any of these functions. To override the function `theme_something()`, define the function `mytheme_something()` in your `.theme` file. This function should have the same definition as the original. It is easiest to start with Drupal's function, and apply your changes there: many theme functions contain code logic within them. To avoid problems when upgrading Drupal in the future, it is best to mark the changes between the original Drupal function and your customized version. That way, you can reapply to your

customizations if the original was changed.

Aside from theme functions, there is one function that you need to include, called `mytheme_features()`. This function should return an array of strings, marking the features your theme supports (e.g. search box, logo, mission statement, ...). The theme system will provide toggles and settings for these features in the administration section. In your code, you can retrieve the value of these settings though `theme_get_setting()`. If you are planning on releasing your theme to the public, it is advised to implement all Drupal features, so others can customize your theme.

Available features are:

| | |
|-----------------------------|--|
| logo | A logo can be used. The theme should check the settings <code>default_logo</code> (boolean) and <code>logo_path</code> (string). |
| toggle_logo | The logo can be turned on/off |
| toggle_name | The site name can be turned on/off |
| toggle_search | The search box can be turned on/off |
| toggle_slogan | The site slogan can be turned on/off |
| toggle_mission | The mission statement can be turned on/off |
| toggle_primary_links | The primary navigation bar can be turned on/off/ |
| toggle_secondary_links | The secondary navigation bar can be turned on/off |
| toggle_node_user_picture | The theme can optionally display user pictures next to nodes |
| toggle_comment_user_picture | The theme can optionally display user pictures next to comments |

Here's the `_features()` function from the standard `chameleon.theme`:

```
<?php
function chameleon_features() {
    return array(
        'logo',
        'toggle_name',
        'toggle_slogan',
        'toggle_primary_links',
        'toggle_secondary_links');
}
?>
```

Note that unlike templates and styles, themes are tied to their directory name. If you want to clone a PHP theme, you need to rename its directory, its .theme file and its functions inside the .theme file.

Theme coding conventions

This theme coding style guide is based on the cvs log message of a developer sick of fixing strange spacing and indentation.

Theme authors should take care to consistently treat spacing and indentation in their code. Just as we have rules for indenting code - because this makes it easier to understand and maintain - there are basic rules for themes and included HTML files:

- Each level of indentation adds 2 spaces
- Match the indentation of (long) opening and closing block html tags
- Distinguish between PHP and HTML indentation. Not

```
function header($title = "") {
<?PHP
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
...
```

but

```
function header($title = "") {
<?PHP
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
...
```

This not only saves the superfluous leading spaces, but also makes it much easier to find matching opening and closing tags defined in functions with different indentation.

- Prefer PHP in HTML to HTML in PHP.

Example: not

```
<?php
function node($node, $main = 0) {
print "\n<!-- node: $node->title -->\n";
print "<div class=\"$nodetitle\">$node->title</div>";
print "<div class=\"$nodebody\"><span
class=\"$nodedate\">" . $this->links(
array(format_name($node),
```

```

format_date($node->created, "small"), "&nbsp;" )
) . "</span>";
?>

but

function node($node, $main = 0) {
<?PHP
    <!-- node: "<?php print $node->title; ?>" -->
        <div class="nodetitle"><?php print $node->title;
?></div>
        <div class="nodebody">
            <span class="nodedate"><?php print $this->links(
array(format_name($node), format_date($node->created,
"small"), "&nbsp;" ) ); ?></span>

```

After all, PHP is a HTML embedded scripting language - and not the other way around.

Updating your themes

As Drupal develops with each release it becomes necessary to update themes to take advantage of new features and stay functional with Drupal's theme system.

Converting 3.0 themes to 4.0

Required changes

Changes in class definition

Theme class definition uses now a different syntax:
Instead

```
class Theme extends BaseTheme {
```

you should use

```
class Theme_themename extends BaseTheme {
```

where *themename* is name of your theme in lowercase.

Changes in function header()

- Function header() takes now an optional parameter \$title.
Instead

```
function header() {
```

you should use

```
function header($title = "") {
```

- Previously all pages in Drupal site had the fixed page title: *sitename - site slogan*. Now the page title can be dynamic - for example when displaying single node, the page title can be *node title - sitename*. So, instead

```
print variable_get("site_name", "drupal") . " - ". variable_get("site_slogan", "");
```

you should use a more complex syntax:

```
if ($title) {
  print $title . " - ". variable_get("site_name", "drupal");
}
else {
  print variable_get("site_name", "drupal") . " - ". variable_get("site_slogan", "");
}
```

or if you want to use compact version of the same construction:

```
print $title ? $title . " - ". variable_get("site_name", "drupal") :
variable_get("site_name", "drupal") . " - ". variable_get("site_slogan", "");
```

This piece of code checks if \$title is present. If yes, it outputs \$title and site name, if not, it outputs site name and slogan.

- If you used theme_account() function (what outputs login/membership box) in header(), please remove it. Login box placement is controlled in *Administration > blocks* page from now on and theme_account() is no longer used.

Changes in function node()

- format_name() accepts now parameter \$node, not \$node->name. Also \$node->timestamp is replaced with \$node->created. So, instead

```
print strtr(t("Submitted by %a on %b"), array("%a" => format_name($node->name), "%b" => format_date($node->timestamp)));
```

you should use

```
print strtr(t("Submitted by %a on %b"), array("%a" => format_name($node), "%b" => format_date($node->created)));
```

- node_index() is no longer used because Drupal 4.0 has more sophisticated classification system than Drupal 3.0 meta tags. So instead plain simple

```
print node_index($node);
```

you have to use

```
$terms = array();

if (function_exists("taxonomy_node_get_terms")) {
    foreach (taxonomy_node_get_terms($node->nid) as $term) {
        $terms[] = l($term->name, array("or" => $term->tid), "index");
    }
}
print $this->links($terms);
```

- Function `link_node()` accepts an optional parameter `$main`. Instead

```
if ($main) {
    print $this->links(link_node($node));
}
```

you should use

```
if ($links = link_node($node, $main)) {
    print $this->links($links);
} </li>
</ul>
<h5>Changes in function comment()</h5>
<ul><li>format_name() accepts now parameter $comment, not $comment->name. Instead
<pre> print strtr(t("Submitted by %a on %b"), array("%a" => format_name($comment->name), "%b" => format_date($comment->timestamp)));

```

you should use

```
print strtr(t("Submitted by %a on %b"), array("%a" => format_name($comment), "%b" => format_date($comment->timestamp)));
```

Changes in function footer()

- If you used `theme_account()` function (what outputs login/membership box) in `footer()` function, please remove it. Login box placement is controlled in *Administration > blocks* page from now on and `theme_account()` is no longer used.

Optional changes

New function: system()

- Optionally theme can have a `system()` function what provides info about theme and its author:

```
function system($field) {
    $system["name"] = "theme name";
    $system["author"] = "author name";
    $system["description"] = "description of the theme";
    return $system[$field];
}
```

Converting 4.0 themes to 4.1

Required changes

There is no required changes, all Drupal 4.0 themes should also work in Drupal 4.1

Optional changes

theme_head

Insert a function `theme_head()` inside your theme, right after the HTML's `<head>` tag:

```
<html>
  <head>
    <?php print theme_head(); ?>
    ...
  </head>
```

This change allows modules to incorporate custom markup inside `<head>`; `</head>` tags such as Javascript, `<meta>`; tags, CSS and more.

Converting 4.1 themes to 4.2

Required changes

Add a `theme_onload_attribute()` to a `<body>` tag:

```
<body <?php print theme_onload_attribute(); ?> >
```

Optional changes

Take advantage of `settings()` hook

Themes can now populate settings to administration pages using the function `themename_settings()`. Example:

```
function mytheme_settings() {
  $output = form_select("Sidebar placement", "mytheme_sidebar",
    variable_get("mytheme_sidebar", "right"),
    array(
      "none" => t("No sidebars"),
      "left" => t("Sidebar on the left"),
      "right" => t("Sidebar on the right"));
}
```

Direct your site logo to index.php

If your theme has the logo and you have made it to link `` or even `<a href="<?php print path_uri();?>">`, then please replace these instances with a simple ``.

One additional change may be needed. Using a custom theme adapted from a generic one, the original node function has the following code:

```
<?php
$terms = array();
```

```

if (function_exists("taxonomy_node_get_terms")) {
  if ($terms = taxonomy_node_get_terms($node->nid)) {
    $taxlinks = array();
    foreach ($terms as $term) {
      $taxlinks[ ] = l($term->name, array(
        $term->tid), "index");
    }
    $taxo = $this->links($taxlinks);
  }
}
?>

```

Which gives an error on the index page after upgrading from 4.1 to 4.2 and contains invalid URLs. Replacing the above code with this fixes the error.

```

<?php
$terms = array();
if (module_exist("taxonomy")) {
  $terms = taxonomy_link("taxonomy terms", $node);
}
$taxo = $this->links($terms);
?>

```

Converting 4.2 themes to 4.3

No changes are required :)

A few more CSS classes are available to you if you wish to use them. A non-exhaustive list is

- **read-more**: affects the formatting of the 'read more' link
- **cell-highlight**: affects the cell in the table header which is currently the sort key. this cell also has an image which you can override in your theme->image directory (most images are overridable in this way).

Converting 4.3 themes to 4.4

For more information on how the interaction between themes and modules has changed, see converting 4.3 modules to 4.4.

- The theme system is no longer built on PHP's object model. The BaseTheme class is no more and, as such, you no longer have to use a class for your theme. Instead, a theme is a collection of functions. This will make Drupal theme development feel much the same as Drupal module development. Prefix your theme function with your theme's name. Examples:
`mytheme_page()`, `mytheme_comment()`, `mytheme_node()`.
- `mytheme::system()` (or in the new parlance, `mytheme_system()`) is no longer used. The theme description used on the theme administration page

should instead be returned by a new function called `mytheme_help()`. This function follows the same semantics as the regular `module_help` hook:

```
<?php
function mytheme_help($section) {
    switch ($section) {
        case 'admin/system/themes#description':
            return t("A description of mytheme");
    }
}
?>
```

- All theme functions now return their output instead of printing them to the user. There should be no print or echo statements in your theme.
- The `mytheme_header()` and `mytheme_footer()` functions are no longer used, a `mytheme_page()` function is introduced instead.

```
<?php
function mytheme_page($content, $title = NULL, $breadcrumb
= NULL) {
    if (isset($title)) {
        drupal_set_title($title);
    }
    if (isset($breadcrumb)) {
        drupal_set_breadcrumb($breadcrumb);
    }
    ...
}
?>
```

This function should return the HTML code for the full page, including the header, footer and sidebars (if any). Note that it is important to set the title and the breadcrumbs for Drupal with the setter functions as suggested above, instead of just using the values provided as parameters. This way modules acting on the title or breadcrumb values can use the real value when generating blocks for example.

- Themes now have the responsibility of placing the title, breadcrumb trail, status messages, and help text for each page. This gives them the flexibility to, for example, place the breadcrumb trail above the title or in the footer. It is now expected that `mytheme_page()` will return these elements. The `page` theme function should override the title and breadcrumb trail retrieved from Drupal, in case some explicit value is provided in the function parameters (see above). A theme can obtain the values set before by calling the functions `drupal_get_title()`, `drupal_get_messages()`, `menu_get_active_help()`, and `drupal_get_breadcrumb()`. The breadcrumb trail is returned from the latter function as an array of links; it can be formatted into a string by using `theme("breadcrumb", drupal_get_breadcrumb())`. Most themes use the following new code-snippet in their `page` function:

```
<?php
if ($title = drupal_get_title()) {
    $output .= theme("breadcrumb", drupal_get_breadcrumb());
    $output .= "<h2>$title</h2>";
```

```

}
if ($help = menu_get_active_help()) {
  $output .= "<div class=\"$help\">$help</div><hr />";
}
foreach (drupal_get_messages() as $message) {
  list($message, $type) = $message;
  $output .= "<strong>" . t("Status") . "</strong>:$message<hr />";
}
?>

```

- The `_head()` hook is eliminated and replaced with the `drupal_set_html_head()` and `drupal_get_html_head()` functions, therefore the HTML head part should include the return value of `drupal_get_html_head()` instead of the return value of `theme("head")`.
- The `theme_node()` function takes an extra parameter now, `$page`, that indicates to the theme whether to display the node as a standalone page or not. If `$page` is true, then the title of the node should not be printed, as it will already have been printed by `theme_page`. Also note that the node body will only be filtered with the configured filters if the node page is displayed. Otherwise only the teaser will be filtered for performance reasons. Example:

```

<?php
function mytheme_node($node, $main = 0, $page = 0) {
  if (!$page) {
    $output = "<h2>" . $node->title . "</h2>";
  }
  if ($main && $node->teaser) {
    $output .= "<div>" . $node->teaser . "</div>";
  }
  else {
    $output .= "<div>" . $node->body . "</div>";
  }
  return $output;
}
?>

```

- To improve block themeability, `theme_block()` has been changed. The old

```
function theme_block($subject, $content, $region = "main")
```

has become

```
function theme_block($block)
```

with `$block` being an object containing `$block->subject`, `$block->content`, etc. See the doxygen doc for details and for how you can style blocks with CSS.

- Also, `theme_blocks()` has been improved to allow themes to hook into (change) the blocks before outputting them. See this cvs log message for details.

foo bar

Converting 4.4 themes to 4.5

Note: the theme system changed significantly in 4.5. Make sure you read through this entire guide, as an outdated theme will prevent you from accessing vital parts of your site.

Directory structure

Templates are now seen as themes unto themselves, rather than hiding behind their template engine. Template engines now reside in subdirectories of themes/engines, while templates simply are placed in subdirectories of themes. Template engines compatible with Drupal 4.5 will identify templates based on their filename and send the appropriate listings to the theme system.

For xtemplate templates, your template must be named `xtemplate.xtmp1`, and your default stylesheet must be named `style.css` (as mentioned below in the "Styles" section).

For example, the old Xtemplate pushbutton template has moved from themes/xtemplate/pushbutton to themes/pushbutton.

Tabs (a.k.a. Local Tasks)

Drupal now separates out menu items that are "local tasks"; functions to be performed on the current location. By default, these are rendered as a set of tabs. Themes are responsible for printing these. A typical location is below the page title, so that

```
<?php
if ($title = drupal_get_title()) {
  $output .= theme("breadcrumb", drupal_get_breadcrumb());
  $output .= "<h2>$title</h2>";
}
if ($help = menu_get_active_help()) {
  $output .= "<small>$help</small><hr />";
}
?>
becomes
<?php
if ($title = drupal_get_title()) {
  $output .= theme("breadcrumb", drupal_get_breadcrumb());
```

```

    $output .= "<h2>$title</h2>";
}
if ($tabs = theme('menu_local_tasks')) {
    $output .= $tabs;
}
if ($help = menu_get_active_help()) {
    $output .= "<small>$help</small><hr />";
}
?>

```

For xtemplate templates, *Before:*

```

<!-- BEGIN: title -->
{breadcrumb}
<h1 class="title">{title}</h1>
<!-- END: title -->

```

After:

```

<!-- BEGIN: title -->
{breadcrumb}
<h1 class="title">{title}</h1>
<!-- BEGIN: tabs -->
<div class="tabs">{tabs}</div>
<!-- END: tabs -->
<!-- END: title -->

```

Status Messages

The `theme_page` function is no longer responsible for rendering each status message. Instead, we now use the `theme_status_messages()` function. *Before:*

```

<?php
foreach (drupal_get_messages() as $message) {
    list($message, $type) = $message;
    $output .= "<strong>". t("Status") . "</strong>:
$message<hr />";
}
?>

```

After:

```

<?php
$output .= theme_status_messages();
?>

```

Static vs. Sticky

In Drupal 4.5, "static" posts have been renamed as "sticky" posts. If your theme uses special styling for this type of post, you'll want to change any references from "static" to "sticky".

Avatar vs. User Picture

In Drupal 4.5, "avatars" have been renamed to "user pictures". Additionally, the method by which themes display avatars has changed. Themes now call `theme_user_picture`, which returns the appropriate image and link HTML.

Before:

```
<?php
    if (module_exist("profile") &&
variable_get("theme_avatar_node", 0)) {
    $avatar = $node->profile_avatar;
    if (empty($avatar) || !file_exists($avatar)) {
        $avatar = variable_get("theme_avatar_default", "");
    }
    else {
        $avatar = file_create_url($avatar);
    }
    if ($avatar) {
        $avatar = "<img src=\"$avatar\" alt=\"" . t("%user's
avatar", array("%user" => $node->name ? $node->name :
t(variable_get("anonymous", "Anonymous")))) . "\" />";
        if ($node->uid) {
            $avatar = l($avatar, "user/view/$node->uid",
array("title" => t("View user profile.")));
        }
        $output .= $avatar;
    }
}
?>
```

After:

```
<?php
    $output .= theme('user_picture', $node);
?>
```

For xtemplate templates, simply replace:

```
<!-- BEGIN: avatar -->
<div class="avatar">{avatar}</div>
<!-- END: avatar -->
```

with:

```
<!-- BEGIN: picture -->
{picture}
<!-- END: picture -->
```

Theme Screenshots

The new theme selector looks for a screenshot of each theme with the filename `screenshot.png` in each directory. Screenshots are optional and themes without screenshots will simply display "no screenshot" on theme selection pages. To create a screenshot which matches those in core, follow these instructions:

1. Log in as administrator user.
2. Enable the following modules, for some extra menu items:
aggregator, blog, node, page, story, tracker
3. Create the following story node:
 - title:** Donec felis eros, blandit non.
 - body:** Morbi id lacus. Etiam malesuada diam ut libero. Sed blandit, justo nec euismod laoreet, nunc nulla iaculis elit, vitae. Donec dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus vestibulum felis nec libero. Duis lobortis. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nunc venenatis pretium magna. Donec dictum ultrices massa. Donec vestibulum porttitor purus. Mauris nibh ligula, porta non, porttitor sed, fermentum id, dolor. Donec eu lectus et elit porttitor rutrum. Aenean justo. Phasellus augue tortor, mattis nonummy, aliquam euismod, cursus eget, ipsum. Sed ultricies bibendum ante. Maecenas rhoncus tincidunt eros.
4. Look at the node, and make sure the tabs are visible. Take a screenshot.
5. Cut out a piece about 420x254 resized to 150x90 (35% zoom). Try to show useful page elements (menu, tabs, title, links).
6. Applied a plain 'sharpen' filter to the thumbnail.
7. Save as "screenshot.png" in theme (or style) directory.

Centralized Theme Configuration

The theme system now has the ability to store certain common configuration items for each theme. However, some themes may not wish to utilize all of these settings, so a `theme_features` hook has been introduced. In each theme / theme engine, this function should return an array of settings which the theme supports. To implement each of these functions, themes / theme engines should call the `theme_get_setting` function, which will return data regarding the administrator's setting for this particular theme. If there are no settings for the current theme, global values will be returned. Below is a table of values for the `_features` hook, a description of their function, and a code snippet of the appropriate `theme_get_settings` call.

| <code>_features</code> hook value | Description | <code>theme_get_settings</code> call |
|-----------------------------------|-------------|--------------------------------------|
|-----------------------------------|-------------|--------------------------------------|

| | | |
|-------------------------------|--|---|
| 'logo' | theme allows customization of site logo | <?php if (\$logo = theme_get_setting('logo')) { \$output .= " "; } ?> |
| 'toggle_name' | theme allows site name to be switched on/off | <?php if (theme_get_setting('toggle_name')) { \$output .= " <h1 class=\"site-name title\">" . l(variable_get('site_name', 'drupal'), ""). "</h1>"; } ?> |
| 'toggle_search' | theme allows search box to be switched on/off | <?php if (theme_get_setting('toggle_search')) { \$output .= search_form(); } ?> |
| 'toggle_slogan' | theme allows site slogan to be switched on/off | <?php if (theme_get_setting('toggle_slogan')) { \$output .= " <div class=\"site-slogan\">" . variable_get('site_slogan', '') . "</div>"; } ?> |
| 'toggle_mission' | theme allows site mission to be switched on/off | <?php if (\$mission = theme_get_setting('mission')) { \$output .= \$mission; } ?> |
| 'toggle_primary_links' | theme allows primary links to be customized | <?php \$output .= theme_get_setting('primary_links'); ?> |
| 'toggle_secondary_links' | theme allows secondary links to be customized | <?php \$output .= theme_get_setting('secondary_links'); ?> |
| 'toggle_node_user_picture' | theme allows node user pictures to be switched on/off | <?php if (theme_get_setting('toggle_node_user_picture') && \$picture = theme('user_picture', \$node)) { \$output .= \$picture; } ?> |
| 'toggle_comment_user_picture' | theme allows comment user pictures to be switched on/off | <?php if (theme_get_setting('toggle_comment_user_picture') && \$picture = theme('user_picture', \$comment)) { \$output .= \$picture; } ?> |
| N/A (Global Setting) | Allow admin to specify which node types should display "Submitted by..." message | <?php \$output .= theme_get_setting("toggle_node_info_\$node->type") ? t("Submitted by %a on %b.", array("%a" => format_name(\$node), "%b" => format_date(\$node->created))) : ''; ?> |

Note that all of these settings are optional, but recommended.

Theme-specific settings are still possible as well. They are still read from the `theme_settings`, but are now placed in a group on the appropriate theme's tab, rather than on a separate page.

Styles

The theme system now allows for switching between different "styles" for each theme. Each "style" is defined by a `style.css` file in a subdirectory of the theme. In order to accomplish this style switching, themes should add a call to `theme_get_styles()` within their `<head>` block. For example:

```
<?php
$output .= drupal_get_html_head();
$output .= " &lt;link rel=\"stylesheet\" type=\"text/css\" href=\"themes/chameleon/common.css\" /&gt;\n";
$output .= theme_get_styles();
$output .= "&lt;/head&gt;";
?>
```

Notice how the reference to `common.css` is listed before `theme_get_styles()`. This allows individual styles to override your common CSS rules (if you use any).

The "default" style for each theme (the stylesheet in which you define color scheme and other general presentation items) should be renamed to `style.css` and placed in your theme directory. You should also remove any references to it from your theme or template. Drupal will reference it in `theme_get_styles()`. (If the default style is selected)

For xtemplate themes, you need to add the `{styles}` tag add the end of your `<head>` section.

_help hook

The `theme_help` hook is no longer used. It can be removed if desired.

References to `comment_referer_load()` should be switched to `comment_node_url()`

Converting 4.5 themes to 4.6

Search form

If your theme implements a search form, it needs to be altered. The search box `<input>` tag should have the `name` attribute set to `edit[keys]` rather than `keys`.

Node links

Node links no longer use the `link_node()` function, but instead are passed as an array in `$node->links`. PHP-based themes will need to be updated to pass this array through `theme('links')`. Template-based themes shouldn't need any changes.

Pages

The function `theme_page()` no longer takes `$title` or `$breadcrumb` arguments. Remove the two arguments and any special handling of them. All page titles and breadcrumbs are now retrieved using `drupal_get_title()` and `drupal_get_breadcrumb()`.

Node and comment markers

Node and comment markers are not restricted anymore to signal that something is new or that a form element is required. The required form element marker was moved to `theme_form_element()`, while `theme_mark()` was kept to generate content markers. New constants help in deciding on the marker to display: `MARK_NEW` signals new content, `MARK_UPDATED` is for changed or extended content and `MARK_READ` is for read or too old content. Now it is possible to output markers for read content too, and distinguish between new and updated content.

Pager and menu item themeing

Parts of the pager are now themeable themselves. The menu theming was also reorganized, to be easier to add wrappers and theme menu links. If you override any of the `theme_menu...()` functions in your theme or template, compare them to the current versions in `theme.inc` and `menu.inc` to update them.

Text validation changes

Due to some changes in plain-text processing, some parameters which were HTML are now plain-text and vice-versa. If you use a theme engine, you shouldn't need any changes, except if you override extra `theme_` functions yourself.

If you are seeing problems, the best approach is to compare every `theme_` functions that you override with the one from Drupal core. In particular, the menu theme functions (`theme_menu_*`) require changes in the way `l()` is used.

You should also try submitting a node and a comment with HTML tags in the subject. The tags should come out escaped, and should be shown on screen rather than interpreted. If this is not the case, you need to check your `theme_node()` and `theme_comment()` functions.

Finally, page titles should be run through `strip_tags()` when put into the `html <title>` tag in `<head>`. This should only be a problem if you have a `.theme` theme, as the theme engines have all be updated to accomodate this change.

Converting 4.6 themes to HEAD

Table row coloring

The class names for alternating table rows have been changed from light and dark to odd and even.

Theme screenshot guidelines

Every theme for 4.5+ needs a screenshot in the form of a screenshot.png placed in the theme/template/style directory. It is best that screenshots are consistent. The guidelines for core theme screenshots are (starting from a blank Drupal site):

1. Log in as the first user.
2. Enable the following modules, for some extra menu items: *aggregator, blog, node, page, search, story and tracker*.
3. Turn on the features that the theme supports (logo, site name, slogan, search box). Add some primary and secondary links if needed. We suggest "Link 1" "Link 2" "Link 3", you can link them to e.g. "user/1".
4. Set the site name to *Drupal* and slogan to *Community Plumbing*.
5. Create the following story node:

Donec felis eros, blandit non

Morbi id lacus. Etiam malesuada diam ut libero. Sed blandit, justo nec euismod laoreet, nunc nulla iaculis elit, vitae. Donec dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus vestibulum felis nec libero. Duis lobortis. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nunc venenatis pretium magna. Donec dictum ultrices massa. Donec vestibulum porttitor purus. Mauris nibh ligula, porta non, porttitor sed, fermentum id, dolor. Donec eu lectus et elit porttitor rutrum. Aenean justo. Phasellus augue tortor, mattis nonummy, aliquam euismod, cursus eget, ipsum. Sed ultricies bibendum ante. Maecenas rhoncus tincidunt eros.

6. Look at the node, and make sure the tabs are visible. Take a screenshot.
7. Cut out a piece about ~420x254 resized to exactly 150x90 (~35% of the original size). Try to show only useful page elements (menu, tabs, title, links). Don't include browser chrome (toolbar, status bar, scrollbar, etc).
8. Apply a standard 'sharpen' filter to the thumbnail for clarity.
9. Save as a PNG, in palettes colorspace to cut down on size.

Example:



For Drupal.org project thumbnails, use the guidelines above except that:

- You should fill up the site more. For example, add a comment to the story node or add some blocks.
- The screenshot should show the entire page, though still without browser chrome (toolbar, status bar, scrollbar, etc).
- The thumbnail should be 320x200 pixels large. It is best to resize to a width of 320 pixels first, then to crop off the bottom to a height of 200 pixels.
- Try to make your original screenshot have a width of about 1000 pixels, so that the thumbnail is about 30% of the original size.
- Name the screenshot `screenshot-drupal.org.png` or `screenshot-drupal.org.jpg`.
- Try to keep the image small: save as paletted PNG or as a JPEG with 10-20% compression. This reduces the load time of the theme list on Drupal.org. Aim for 15-20KB.
- Include the image in your project's description with class picture:
``
 If you don't have permission to use the `` tag, ask a site maintainer to add your screenshot.
- If your theme is available on the Drupal Theme Garden, turn the thumbnail into link to it:
`<img
...>`

Example:



Theme how-to's

This section collects 'How-to' articles on subjects relevant to theme developers.

Tips for designing themes in Dreamweaver, GoLive etc.

- An Xtemplate-in-GoLive how-to is available at <http://drupal.org/node/6634>

Dreamweaver

Dreamweaver can edit your PHP, template and CSS files just fine, and in some cases (XTemplate, PHPTAL) with a graphical preview.

Find the Dreamweaver configuration files, `MMDocumentTypes.xml` and `Extensions.txt`. where they are depends on what platform is being used - the Find file function is your friend here.

In `Extensions.txt`

Change the line ending in

```
: All Documents  
to include  
TAL,XTMPL,MODULE,THEME,ENGINE  
Likewise, add TAL,XTMPL to the line ending in : HTML files  
and  
MODULE,THEME,ENGINE to the : PHP files line.
```

In `MMDocumentTypes.xml`

you likewise want to add tal and xtmpl file types to the file type descriptions, e.g. change the line begining documenttype id="HTML" to be

```
<documenttype id="HTML" internaltype="HTML"  
winfileextension="htm,html,shtml,shtm,stm,tpl,lasso,xhtml,tal,xtmpl"  
macfileextension="htm,html,shtml,shtm,tpl,lasso,xhtml,tal,xtmpl"  
file="Default.html" writebyteordermark="false">
```

and likewise add .theme, .module and .engine to the appropriate section

```
<documenttype id="PHP_SQL" servermodel="PHP MySQL"  
internaltype="Dynamic"  
winfileextension="php,php3,php4,theme,module,engine"  
macfileextension="php,php3,php4,theme,module,engine"  
file="Default.php" writebyteordermark="false">
```

The local copy of your site can now be edited in Dreamweaver.

Adding your theme to Drupal.org

To add your theme to Drupal.org, it must be GPL. Do not include images or other copyrighted works that you do not want to see re-used or otherwise altered.

Themes are tracked the same way that code is, in the CVS repository. You will need to apply for a CVS account. Once you are approved, you will be able to check your theme into the Drupal CVS repository. Create a project and the download will be created for it automatically.

If you do add your theme, users will likely post suggestions, file bugs, and generally desire that you keep the theme up to date with current versions of Drupal.

Theme snippets repository

Did you write a nice custom template function? Please add a book page below this one. Please remember to document your code well.

Custom login

This creates a little custom login area. If you are logged in, it displays your username and a link to your profile, otherwise it includes a Register and minimal Login area. It was developed for OurMedia.

```
<?php if ($user->uid) : ?>
    Logged in as: <?php print
l($user->name,'user/'.$user->uid); ?> |
<?php print l("logout","logout"); ?>
    <?php else : ?>
        <?php print l("Register","user/register"); ?> | Login:
<form
action="user/login" method="post"><input type="hidden"
name="edit[destination]" value="user" /><input type="text"
maxlength="64" class="form-text" name="edit[name]"
id="edit-name"
size="15" value="" /><input type="password"
class="form-password"
maxlength="64" name="edit[pass]" id="edit-pass" size="15"
value="" /><input type="submit" name="op" value="Log in" /></form>
    <?php endif; ?>
```

Temporarily on the dev wiki until someone can get it posted here.

Customize display of submission information based on node type

You can display different author information and still respect the global "display post information" settings. In this example I wanted flexinode-2 items to be displayed with the usual submission info, but have every other type respect the "display post information" setting. Note that the content type I created, "news", goes by its flexinode name, and not by what I named it. Here is an example snippet for a node.tpl.php:

```
<?php if (theme_get_setting('toggle_node_info_'.  
$node->type)) : ?>  
<?php if ($node->type == 'flexinode-2') :?>  
<div class="info"><?php print $submitted; ?></div>  
<?php elseif($node->type) : ?>  
<div class="info"> by <?php print $name; ?></div>  
<?php endif; ?>  
<?php endif; ?>
```

For my settings this displays:

NEWS NODE TITLE HERE
submitted by me on Tue, 04/26/2005 - 10:55am.

STORY NODE TITLE HERE
by me

PAGE NODE TITLE HERE

because I have set news and story nodes to display the submitted info while page is set to display no submission info at all.

How to display mission on every page?

This seems an easy one, but I can't figure out how to display the site mission on every page, it only shows in the home page... I am using a self-made php template, and I have the following line in page.tpl.php:

Hope you can help me, TIA!

Make images square

Often images are not square -landscape or portrait- which causes rendering problems or just ugly pages. Tables might have different sized rows or columns, inline images look differnt in each post, or rows of images appear horribly cluttered and inconsistent. This function will add padding to your images, to make them appear in a square. But remember that this will override any padding applied to img in your stylesheets.

```

<?php
function themename_image($path, $alt = '', $title = '', $attr
= '', $getsize = true) {
    //always do getimagesize
    list($width, $height, $type, $attr) = @getimagesize($path);
    //get the biggest value.
    if ($width > $height) {
        $padding = round(($width - $height)/2);
        $style_str = ' style="padding:' . $padding . 'px 0;"';
    }
    elseif ($width < $height) {
        $padding = round(($height - $width)/2);
        $style_str = ' style="padding:0 ' . $padding . 'px;"';
    }
    return "<img src=\"$path\" $attr alt=\"$alt\""
    title=\"$title\" $style_str />";
}
?>

```

Overriding drupal.css; two approaches

There are two methods to removing drupal.css from your theme in phptemplate.

The first cuts out the link from the \$head variable. In page.tpl.php, replace your

```

<?php
print $head
?>

```

with (remove the space in 'style'):

```

<?php
print str_replace('<style type="text/css"
media="all">@import "misc/drupal.css";</style>', '', $head);
?>

```

The other method requires overriding the stylesheet import themable function. Simply add this to your theme's template.php file:

```

<?php
function phptemplate_stylesheet_import($stylesheet, $media =
'all') {
    if ($stylesheet != 'drupal/misc.css') return
    theme_stylesheet_import($stylesheet, $media);
}
?>

```

Documentation writer's guide

Drupal documentation for the Drupal handbooks and the admin/help within each Drupal installation is collaboratively maintained on drupal.org by the documentation team. Documentation team members additionally coordinate the creation of marketing materials, the Drupal Newsletter, and other document efforts relevant to the Drupal community.

Getting involved in documentation projects

Anyone may participate in Drupal documentation development as a member of the documentation team:

- Begin by subscribing to and joining in discussions on the drupal-docs mailing list. Anyone can also peruse list archives to get a feel for recent documentation team issues.
- Become acquainted with and/or volunteer for one or more of the Drupal documentation project teams which handle specific documentation needs for the Drupal community.
- Browse open issues in the Documentation project

Requests for new documentation and reporting documentation problems

Ideas for new projects, requests for new documentation, and reports of documentation problems should be submitted as issues to the Documentation project.

Creating or updating pages in the Drupal handbooks

Everyone in the Drupal community is invited to submit new pages or edit existing pages in the Drupal handbook.

- Please review first the authoring guidelines section for handbook pages.
- Any drupal.org site member may create book pages and edit book pages which they have created. These pages will then enter a moderation queue where they will either be
 - approved and published
 - remanded for further revision and remain in moderation
 - rejected
- Since regular drupal.org site members cannot edit book pages which they did not create, additions to or corrections of other handbook pages should be submitted as an issue to the Documentation project.

- Members of the documentation team who are site maintainers may choose to post new pages and edited pages directly, avoiding the moderation queue.
- Rough drafts of new pages or edited pages which are not production ready--i.e., need feedback and/or additional development--should first be submitted as an issue to the Documentation project.

How to add a page to the Handbook

It's very easy to add a page. When you feel like writing a piece of documentation about a problem that hasn't already been addressed, just click on click "create book page" link in your login menu (you have to be logged in to see it) and type away in the *Body*.

You will have to set the correct parent to page, select it under *Parent* drop-down menu.

Optionally you can add a *log message* if you please, in which you explain why you wrote the documentation.

Finally you give your piece of documentation a *weight*. The pages with a less heavy weight will stay at the top within the parent section while pages with heavy weights will sink deeper. Then preview and finally commit.

Once you submit your documentation it will be placed in the moderation queue where one of the site maintainers will review your document and eventually post the page if it meets with the guidelines.

Authoring guidelines

All pages, submissions, and edits in the Handbook must follow the following guidelines. Updates to the Handbook submitted to the moderation queue will not be published until they follow these guidelines. Numbers on the left are for reference.

| 1. Titles | |
|-----------|---|
| 1a | Use short descriptive titles to label a handbook entry. Avoid redundancy; The words "Drupal" and "Handbook" should rarely be used in titles |
| 1b | Capitalize only the first letter of a title unless using a proper noun. <i>e.g.</i> Installing modules; Designing themes in Dreamweaver |
| 1c | Use "HOWTO: {title}" for documentation pages that function as howto-style tutorials. |

2. Writing style

- | | |
|----|--|
| 2a | Use American spelling. For example, color, not colour. |
| 2b | Do not use 1st person pronouns: I, my, we, our, etc. |
| 2c | Don't be wordy or colloquial. |
| 2d | Avoid creating new documentation for topics already covered in the Handbook. |
| 2e | SPELL CHECK. |

3. Text formatting

- | | |
|----|--|
| 3a | When describing how to get to a specific user interface option (e.g. the add vocabulary option in the categories section of the administer screen) demarcate the path needed to access the option using this format: destination (<i>path > to > item > destination</i>) Which will yield text that looks like this: <i>destination (path > to > item > destination)</i> |
| 3b | Do not abbreviate words in the navigation path (e.g. use administer instead of admin) |
| 3c | Menu items can change places now. Always include the Drupal path for any user interface option. |

4. HTML and code formatting

- | | |
|----|---|
| 4a | Use ordered lists for step by step instructions: |
| 4b | Enclose HTML code samples within <code> tags. |
| 4c | Enclose PHP code samples within <?php and ?> tags. |

| 5. Linking | |
|-------------------|---|
| 5a | Use title tags. See below for examples. |
| 5b | Use relative links where possible: Bad <pre></pre> Good <pre></pre> |
| 5c | Links should be embedded within normal descriptive body text: Bad I like cat stomachs. To learn more about them click here Good My favorite organ in a cat is the stomach |
| 5d | When describing an example URL always use 'www.example.com' as the domain name. |
| 5e | Include links to relevant or related documentation and/or forum posts as much as possible. You only need to link to each relevant item once. |

Adding screenshots

Screenshots on Drupal.org must follow these standards:

Format: PNG-8

PNG is preferred because of its lossless compression and small file size. JPEG is not suitable - it leaves artifacts and its 24-bit color depth is unnecessary when dealing with screenshots. GIF is not used because it is an outdated format.

Size:

It is preferred that image size do not exceed 700 x 700 pixel size.

Browser window

All screenshots must include the entire browser window e.g. the title bar and scrollbars. If your operating system or capturing utility supports this, try to capture only the browser window, not the entire desktop (otherwise you will have to crop it in an image utility manually).

When preparing to capture, remove all distracting items from your browser application - sidebars, tabs, toolbars, Google bars, custom links, etc. Keep only the URL address bar and make sure it is long enough that the current URL won't be hidden.

Windows XP

If you can, please turn off ClearType font smoothing when taking screenshots under XP. Most people don't have this feature and it increases the filesize somewhat.

For reference, see Screenshots in the GNOME Documentation.

Avoid all embedded headings

Any use of H1, H2, ... H6 inside of a book page is a sign of a structural problem. These should *never* appear. The presence of these elements implies:

- It will be impossible for others to add or re-arrange sections unless they have edit permissions for a page.
- Individual authors can hijack the apparent (presentational) structure of a book by arbitrarily beginning new chapters, for example
- There is no guarantee of coherence (hierarchy) in the presentation of the book
- The structure of the document is not accessible to the navigational elements presented by Drupal - sections will disappear from the book navigation view, forward/up/next links, etc.

If you feel the urge to include them in your book page, one or more of the following is probably true:

- Your page is too long. If readers need to scroll through a long page, they will probably need the navigational clues that headings provide - but they will in all cases be better served by having the page divided into true subsections via child nodes.
- You are trying to present some kind of list. Try using `<dl>`, `<dt>`, `<dd>` instead.
- You are trying to achieve visual impact. Visual impact is OK, though it's often not done well. If you must, do it with style (pun intended).
- Your page is essentially just a container for subsections - try pushing content up into the parent or down into children and eliminate the page entirely.

HOWTO: Writing admin/help documentation

Most of the administration help documentation which resides in each Drupal installation as of Drupal 4.? is maintained in the Drupal modules and features section of the handbook as the introductory page for help on each core and contributed module. During feature freeze of each new major Drupal release, the Drupal documentation team updates the module page in the handbook to the latest Drupal version. It is then rolled into Drupal core.

Resources for getting started

1. First, familiarize yourself with the general Drupal handbook authoring guidelines. You should write documentation with these guidelines in mind since all submitted documentation will be reviewed for compliance with these guidelines.
2. The documentation team and the development community have decided that the module pages and administration help documentation needs short, scannable documentation. If you would like to know more about why users scan web documentation, read Why Scanning.
3. Familiarize yourself with the format covered in this document by reviewing existing module pages in Drupal modules and features section.

Guidelines for structure, content, and links

Structure

- The module documentation should be brief without repetition: 2 paragraphs and a list of links.
- In no more than 2 to 3 sentences, the first paragraph should clearly and succinctly describe the module through its benefits to users.
- The second paragraph should provide an explanation of module features that will help users to accomplish the most common tasks.
- The text should be followed by a list of complete sentences that describe a task and link to the page where administrators can accomplish that task.
- The phrase "For more information, read the configuration and customization handbook modulename page" will be automatically added to the administration help when these pages are extracted from the handbook and added to the module code.

Content

- For consistency, the first sentence of the first paragraph will include the module name.
- Use the term post instead of node.

Links to common tasks

- The links section should be easily scannable so that users can quickly navigate to relevant interfaces.
- The list of links should begin with the phrase "You can" and be followed by an unordered list of complete sentences that describe a task and link to the page where administrators can accomplish that task. If the list has only one item it should be a single paragraph beginning with 'You can'.
- Admin help links should be context sensitive into the administration interfaces. Provide the direct link in the links section and label it appropriately, followed by path instructions. For example, "administer blog at

- administer >> content >> configure >> content types >> personal blog entry ."
- If the module requires access permissions to be set, provide a link to the access permissions page. Ed. this is still under discussion.
 - If the module is a content type, provide a link to create content type.
 - If the content module has page that displays all content of that type, provide a link to that page.
 - If the content type has configurable submission guidelines then a link should be provided to it. Ed-example is needed.
 - Internal links must be extracted and replaced with a variable for the t() function. These links must be in the format href = "/path/path" in order for these links to be extractacted by scripts.

Modifying the handbook modules pages for inclusion in Drupal core

- Include a link referring to the original handbook page after the links section using the following format : "For more information, read the configuration and customization handbook module page."
- Submit the finalized text as issues against the "help" component of the Drupal project. Announce this on docs/drupal-dev requesting that developers roll patches. Of course, issue follow-ups can also be used to do tweaking of documentation as well.
- The t function should enclose the text with single quotes. All single quotes in the content need to be escaped with a backslash.
- All internal URLs must be replaced with variables %variablename, and be added to an array for the t function.
- Do not use single or double quotes for emphasis as they are difficult to parse for inserting into the code.

Spelling and capitalization

This page is a supplement to the Authoring guidelines. The goal is to achieve a greater consistency and quality in Drupal documentation. This page is under construction. To contribute to this page, please post a message on the Drupal Docs mailing list.

- **Drupal, not drupal**
- e-mail, not email
- HTML, not html
- URL, not url
- PHP, not php
- MySQL
- JavaScript

Responsibilities of documentation project teams

- For new projects, documentation teams will create and offer a project proposal to the rest of the documentation community.
- Documentation teams will maintain a sub-page of the the Drupal documentation project teams page listing current priorities and todo projects. Active project specs will either reside there as sub-pages (for larger ongoing projects) or provide a link to the project from the main project team page.
- Documentation team members with site maintainer permissions should regularly
 - check recent updates to the handbook and review pages needing moderation.
 - assist in updating page updates listed in the Documentation project issues.

Documentation project proposals and specs

Anyone seeking to initiate new documentation projects --project teams and individuals--must submit a project proposal and post and maintain a project spec.

Goals of project proposals

- Allow the documentation community to provide feedback to improve the project.
- Will help to build consensus for projects as part of a vetting process.
- Can help to gain volunteers for the project.
- Makes the project more transparent/open for the rest of the Drupal community.
- Is necessary to gain approval for the project.

Goals of project specs

- Helps to clearly define objectives and guidelines for project participants.
- Allows others, less involved in the project planning, to more easily assist in the project.
- Can serve as resource for developing future, similar projects.
- Makes the project more transparent/open for the rest of the Drupal community.
- Is necessary to gain approval for the project.

Project proposals and specs should attempt to address each of the following:

- Objectives/goals of the project
- Intended audience of documentation

- Relevant issues, i.e.
 - those addressed by the project
 - those that will not be addressed by the project
 - any still needing to be addressed by the project
- Timeline or deadlines (if applicable)
- Workflow description
- Guidelines and examples for documentation construction (if applicable)
- List of main participants and/or organizers
- Requests for specific feedback and/or additional project participants (if applicable)

Submitting and maintaining the project proposal and spec:

- The proposal will be posted as a new issue to the Documentation project.
- Feedback from community members should be submitted by posting to the specific project issue.
- Proposal submitters will revise the proposal based on community feedback into a project spec.
- During the project, the project spec should be maintained as changes occur in the project (i.e., new documentation construction guidelines, changes in deadlines, etc.).
- For project teams, active project specs will either reside as a sub-page of the Drupal documentation project teams page for the (for larger ongoing projects) or provide a link to the project from the main project team page.

Translator's guide

This is the Drupal translator's guide. It will cover most aspects of translating Drupal's user interface. It will not cover the use of the various programs that can be used to do a translation. These programs are usually quite well documented.

As of version 4.5.0, Drupal includes an extended locale.module that enables you to share translations through the use of PO files. PO files are files containing translations as used by the GNU gettext program.

User contributed PO files for various languages can be found on the download page.

If your language is not present, you might want to start a translation yourself. If this is the case, please download the Drupal POT translation templates. You can get a PO file editor and start translating.

You should translate the individual PO files (per module) rather than one big file. The individual files are automatically packaged into one large file per language in the CVS repository, which is what others will download from this site.

Once you have completed a reasonable part of the translation, create an issue on the *Translation templates* project and upload your PO files. Some helpful developer will then come by and put them in CVS for you. If you have write access to the contrib CVS you can commit your files yourself. In any case a project for your translation will be created, you will be made the maintainer, and your translation becomes available on the download page

Translation templates

Translators should start by downloading the tarball and translating the files to their language of choice.

The translated files should be stored in contrib-cvs/translations/*id* where *id* is the ISO 639 language code. If you don't know your code, ask in drupal-devel.

You should only put the individual translated files in this directory. A script will generate a merged *id.po* file. Make sure to fill out the header section of each file and rename them from .pot to .po.

If you do not have a CVS account, create an issue for this project and attach your files to it.

Note that the Drupal team will not check contributed translations for accuracy or errors.

Programs to use for translation

Recommended PO file editors are (in no particular order):

- XEmacs (with po-mode): runs on Unices with X
- GNU Emacs (with po-mode): runs on Unices
- KBabel: runs on KDE
- poEdit: linux and windows
poEdit does support multiple plural forms since version 1.3.

For Mac OS X there is AquaEmacs and a port of GNU Emacs available using carbon for OS X:

http://www.apple.com/downloads/macosx/unix_open_source/carbonemacspackage.html

also see the Emacs wiki for more usage help and tips:

<http://www.emacswiki.org/cgi-bin/emacs-en/CarbonEmacsPackage>

po-mode is not included, but is easy to add. get it from the GNU gettext distribution.

Be sure to get a recent version for all editors, multiple plural forms are a recent addition to the gettext standard.

Issues using poEdit

poEdit for windows, version 1.3.1 (latest at the moment) doesn't seem to recognize plural forms (if you try to edit a term which has plurals, even if you translate it, it doesn't appear in poedit when you move to an other term, as usual, and even if you save, it doesn't).

Plurals Solution #1

So, if you find a plural term, close poedit, open the file you were translating with a normal text editor (no, not Word...), and search for "plural" in it, you find something similar to this:

```
#: modules/comment.module:187 modules/node.module:89
msgid "1 comment"
msgid_plural "%count comments"
msgstr[0] "1 commento"
msgstr[1] "%count commenti"
```

simply tranlate the text in **msgid** (singular form) into **msgstr[0]**, and the text in **msgid_plural** (plural form) into **msgstr[1]**, save the file, close the editor and return to poedit. Even better, you can do this BEFORE start translating the rest of the file with poedit, translating every occurrence of plural in the same way, in every file, and THEN start using poedit: this way, you will find those strings already translated in poedit, and they don't bother you.

Plurals Solution #2

To use plurals in PO edit you can start with the catalog setting for english and then modify to suit. The syntax is:

```
nplurals=2; plural=(n != 1);
```

which gave me what I needed in Swedish translation of:

```
#: modules/aggregator.module:100;711;722
msgid "1 item"
msgid_plural "items"
msgstr[0] "1 inlÃ¤gg"
msgstr[1] "%count inlÃ¤gg"
```

I tested this in PO Edit 1.3.1 and got the proper GUI response and saved without error.

Plurals Solution #3

The plural forms to use in PO edit under catalog-settings where you see

nplural=INTEGER; plural=EXPRESSION

Only one form:

Some languages only require one single form. There is no distinction between the singular and plural form. An appropriate header entry would look like this:

Plural-Forms: nplurals=1; plural=0;

Languages with this property include:

Finno-Ugric family
Hungarian

Asian family
Japanese, Korean

Turkic/Altaic family
Turkish

Two forms, singular used for one only
This is the form used in most existing programs since it is what English is using. A header entry would look like this:

Plural-Forms: nplurals=2; plural=n != 1;

(Note: this uses the feature of C expressions that boolean expressions have to value zero or one.)

Languages with this property include:

Germanic family

Danish, Dutch, English, German, Norwegian, Swedish

Finno-Ugric family

Estonian, Finnish

Latin/Greek family

Greek

Semitic family

Hebrew

Romanic family

Italian, Portuguese, Spanish

Artificial

Esperanto

Two forms, singular used for zero and one

Exceptional case in the language family. The header entry would be:

Plural-Forms: nplurals=2; plural=n>1;

Languages with this property include:

Romanic family

French, Brazilian Portuguese

Three forms, special case for zero

The header entry would be:

Plural-Forms: nplurals=3; plural=n%10==1 && n%100!=11 ? 0 : n != 0 ? 1 : 2;

Languages with this property include:

Baltic family

Latvian

Three forms, special cases for one and two
 The header entry would be:

```
Plural-Forms: nplurals=3; plural=n==1 ? 0 : n==2 ? 1 : 2;
```

Languages with this property include:

Celtic
 Gaeilge (Irish)

Three forms, special case for numbers ending in 1[2-9]

The header entry would look like this:

```
Plural-Forms: nplurals=3; \
    plural=n%10==1 && n%100!=11 ? 0 : \
        n%10>=2 && (n%100<10 || n%100>=20) ? 1
    : 2;
```

Languages with this property include:

Baltic family
 Lithuanian

Three forms, special cases for numbers ending in 1 and 2, 3, 4, except those ending in 1[1-4]

The header entry would look like this:

```
Plural-Forms: nplurals=3; \
    plural=n%10==1 && n%100!=11 ? 0 : \
        n%10>=2 && n%10<=4 &&
    n%100>=20 ) ? 1 : 2;
```

Languages with this property include:

Slavic family
 Croatian, Czech, Russian, Slovak, Ukrainian

Three forms, special case for one and some numbers ending in 2, 3, or 4

The header entry would look like this:

```
Plural-Forms: nplurals=3; \
    plural=n==1 ? 0 : \
        n%10>=2 && n%10<=4 &&
    n%100>=20 ) ? 1 : 2;
```

Languages with this property include:

Slavic family
Polish

Four forms, special case for one and all numbers ending in 02, 03, or 04
The header entry would look like this:

```
Plural-Forms: nplurals=4; \
plural=n%100==1 ? 0 : n%100==2 ? 1 : n%100==3 || n%100==4 ? 2
: 3;
```

Languages with this property include:

Slavic family
Slovenian

Long plural formulae: Those should not be broken into several lines in the header of the PO file. Drupal expects the formula to be on one line. One could consider this a bug.

I don't think that you can use line breaks in POedit either. The text is fixed to keep from breaking the site layout. But this:

```
nplurals=1; plural=ar;
```

produces an error. The plural form "ar" is not recognized.

Setting up XEmacs with po-mode on Windows

XEmacs has been supported on Windows for a long time and can be downloaded from here: <http://www.xemacs.org/Download/win32>. The po-mode is bundled with XEmacs (no need to get the GNU gettext distribution).

However, you need a MULE-enabled XEmacs binary to edit the UTF-8-encoded PO files and I could not find such a binary for Windows on www.xemacs.org. What I did is this:

1. Install XEmacs 21.4.13 using the Netinstall (<http://www.xemacs.org/Download/win32/setup.exe>)
2. Replace the files installed in the C:\Program Files\XEmacs\XEmacs-21.4.13 directory with those from <http://www.suiyokai.org/tomonori/xemacs/xemacs-i586-pc-win32-21.4.13-mule.tar.gz> (as the filename implies, this is a MULE-enabled XEmacs 21.4.13 binary for Windows)
3. Install the MULE packages in the C:\Program Files\XEmacs\mule-packages directory (these can be downloaded from <ftp://ftp.xemacs.org/xemacs/packages/xemacs-all-mule-packages.tar.gz>).
4. Set the environment variable EMACSPACKAGEPATH with this value:
C:\Program Files\XEmacs\site-packages;C:\Program
Files\XEmacs\mule-packages;C:\Program

Files\XEmacs\xemacs-packages

5. To ensure automatic Unicode detection when opening files, add these lines to your init file (init.el):


```
(require 'un-define)
(set-coding-priority-list '(utf-8))
(set-coding-category-system 'utf-8 'utf-8)
```
6. And finally, add this to automatically enable the po-mode:


```
(require 'po-mode)
```

Yes, this is a bit complicated... Welcome to the wonderful world of XEmacs! :-) If you never used XEmacs before, prepare yourself for a steep learning curve.

BTW, your installation directory does not have to be C:\Program Files\XEmacs. I used this for simplicity in the above instructions.

Translated Drupal information

Some documentation about Drupal (outside of the Drupal interface itself) has also been translated into other languages. Links to such 3rd party translations of external Drupal documentation should live here.

African

This page is for the translation of Drupal Core into Afrikaans. The rest of this text will be in Afrikaans, as that is the purpose of this document.

Vir diegene wat betrokke wil raak by die vertaling:

Stuur 'n e-pos aan Kobus en spesifiseer waarmee jy betrokke sou wou raak. Kobus sal dan met jou in verbinding tree indien jy die nodige besonderhede verskaf het.

Om 'n fout met die vertaling te rapporteer:

Besoek asb. die foutrapporteringsblad

Vir enige ander verwante redes waарoor jy wil kontak:

Stuur 'n e-pos aan Kobus en spesifiseer die presiese rede vir u skakeling.

Russian

Translations and original documentation for Russian Drupal users (still very incomplete, but work is in progress):

- Read about Drupal features in Russian
- Drupal Administrator's Guide - a translation of the original English version
- Drupal Handbook - a translation of the original English version with some rewrites and additions

Visit drupal.ru/docs for a complete list of links to Russian documentation.

Join our efforts to translate Drupal docs into Russian!

Spanish

- Drupal: Manejador de Contenidos y Comunidad Virtual
- Drupal: CaracterÃ-sticas
- Drupal: Manuales

Translation guidelines

To achieve translations that are consistent throughout a whole Drupal site, certain guidelines need to be agreed upon by the translator community for a particular language.

Such guidelines should include a wordlist for words that occur in Drupal's strings. The Ankur Bangla Developer's Guide provides a good example of how this is done on a project unrelated to Drupal. It will be helpfull to not set up a new word lists, but re-use existing ones from an existing translation project.

Other areas which need guidelines will differ from language to language. Please add those guidelines as child pages to this book page.

Translation of contributed modules

Translatable strings from contributed modules are not included in the Drupal core POT files. Module authors can use the *extractor.php* script which comes with the core PO files to generate a POT file on their own. Instructions for running the script can be found in the *README* that comes with the core POT files. The generated POT file should be named as the module, but with a *.pot* extension, e.g. *event.module* gets an *event.pot* file. This file should be placed in a subdirectory *po*. Translations should be added to the same directory. E.g. the *po* subdirectory of *event.module* currently contains the following files: *de.po*, *es.po*, *event.pot*, *he.po*, *hu.po*.

Translators should take care to populate their started translation with the strings from the *general.po* file for their language using *msgmerge*. In this way they can avoid using different translations for terms that occur in both files.

Distributing the translation effort

To facilitate easier handling of a community translation effort, the Drupal POT file is split up into small files that do not contain doubly occurring strings.

All strings that occur more than once in the Drupal core distribution are put into the general.pot file. This ensures that those strings are translated to the same string. Also, files that have ten or less translatable strings will not get their own POT file, but those strings will be appended to the general.pot file.

Of course, some coordination among the project members is still needed to ensure the quality of the translation.

If a language has several options on how to translate some strings, then it is possible to create PO files that only change those strings. An example would be German where you can translate *you* either as *Du* or *Sie* depending on the audience of your site.

Status of the translations

The table below presents an overview of the status of each translation project. This page is updated daily by the package script: it was last updated 1 hour 42 min ago.

Status overview

| Language | cvs | 4.6.0 | 4.5.0 |
|----------|-------------------|-------------------|-------|
| af | 100% (complete) | 100% (complete) | |
| ar | 77% (236 missing) | | |
| bg | 88% (197 missing) | | |
| bn | 3% (1764 missing) | | |
| ca | 64% (409 missing) | 96% (58 missing) | |
| cs | 89% (190 missing) | 89% (190 missing) | |
| da | 100% (complete) | 100% (complete) | |
| de | 76% (380 missing) | 76% (380 missing) | |
| es | 100% (complete) | 100% (complete) | |
| es-la | 0% (256 missing) | | |
| eu | 67% (496 missing) | | |
| fi | 98% (4 missing) | | |

| | | | |
|---------|--------------------|--------------------|--|
| fr | 95% (90 missing) | 100% (complete) | |
| gu | 0% (1825 missing) | | |
| hu | 100% (complete) | 100% (complete) | |
| id | 96% (56 missing) | | |
| it | 94% (91 missing) | 94% (91 missing) | |
| ja | 100% (complete) | 100% (complete) | |
| lt | 67% (543 missing) | | |
| mk | 25% (1148 missing) | | |
| nb | 22% (943 missing) | | |
| nl | 76% (380 missing) | 80% (320 missing) | |
| nno | 26% (1015 missing) | | |
| pl | 17% (1384 missing) | 60% (723 missing) | |
| pt-br | 100% (complete) | 100% (complete) | |
| pt-pt | 100% (complete) | 100% (complete) | |
| ro | 98% (19 missing) | | |
| ru | 71% (358 missing) | 64% (597 missing) | |
| sk | 9% (1436 missing) | 9% (1417 missing) | |
| sq | 39% (687 missing) | | |
| sv | translation broken | | |
| tl-ph | translation broken | | |
| uk | 100% (complete) | | |
| zh-hans | translation broken | translation broken | |
| zh-hant | 99% (1 missing) | 99% (3 missing) | |

Checking your translation status

To see how many of the strings in the PO files you already translated you can try this:

```
for i in *.po; do echo -n "$i: " ; msgfmt --statistics $i ; done
```

Some PO editors already include this feature.

Translations for contributed modules

Contributed modules have POT files in a *po/* subdirectory. Those POT files are created by the author by using the *extractor.php* file that comes with the Drupal POT files. They can also be created by the first translator. The file should be named *module.pot* and stored in the *po/* subdirectory.

Translations to different languages should follow the ISO codes for the languages used for the main translation projects and e.g. be named *pl.po* or *de.po* and also be stored in the *po/* subdirectory..

Make a single file from the loose .po files from CVS

If you want to make a single po file from a CVS folder containing all the small po files, the following commands will do (*nix only). You should execute this, while being in the folder with the .po files.

```
$ msgcat --use-first general.po [^g]*.po | msgattrib --no-fuzzy -o nl.po
```

Off course you should change nl into your own language code.

Recycling old translations

Drupal users with existing translations might want to add those to the translations download page. To do this they first need to export their translation from the localization *manage languages* screen (*export* subtab). Let us assume you have an Italian translation. The above mentioned process will create an *it.po* file for you. To use this file as a basis for a new translation, you treat it as a PO compendium, i.e. a library of pre-translated strings.

This guide assumes a Unix/Linux environment. If you use Windows, check if your PO editor doesn't have a function for this.

We will split the single, large PO file into the smaller files that the Drupal translation Project requires.

First, put the small PO files into a subdirectory *drupal-pot* and your *it.po* file into another one. Then create an empty directory where you want to keep your new small PO files.

Then go to the empty directory and execute the following command from the command line:

```
for i in /path/to/drupal-pot/*.pot ; do msgmerge --compendium  
/path/to/it.po -o 'basename $i .pot'.po /dev/null $i ; done
```

After a while (yes this will take a few minutes) you should have a directory of small PO files that have the matching strings inserted.

Troubleshooting

When doing translations or importing them, several problems can occur. If you think you found a bug in either a translation or in Drupal's locale module, please file bug reports against the project in question.

If you have a more general question you can ask it in the translations forum.

Here we collect some of the more common issues found.

Weird characters or question marks

Symptom: After importing a translation you find all kind of weird characters or question marks on your site.

Solution 1: The translator did not use UTF-8. Drupal is fully UTF-8 aware and expects translations to be supplied in that character set as well. You can change the charset of a PO file using GNU msgconv. Please file a bug against the translation in question.

Solution 2: You do not have the correct font installed to display the language in question.

Marketing resources

This section provides resources for people who want to market drupal. Whether you want to publish a story with a logo on your website, print a leaflet or need some texts for a report, this might be the place to look.

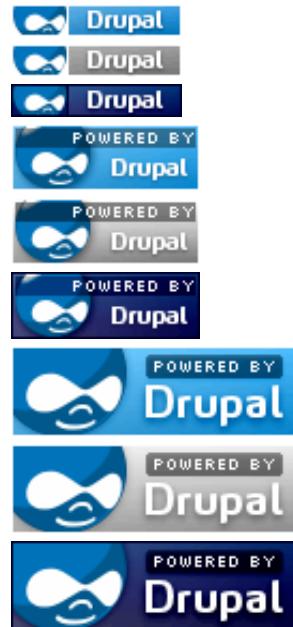
If you are a designer, marketeer or a writer you can be of help here too. If you think you can write a nice text on why people should choose drupal for their it-solutions, or if you have a nice drupal logo you can create a book page here.

Please note that if you have questions concerning any content in this section, you can ask those in the forums. The comments under the bookpages can be used to discuss the contents of that page.

Banners and buttons

If you wish to display the Druplicon on your Drupal website, you can use the following buttons, with a link to drupal.org.

Official buttons



Others



3d small icon



Links

There are several other banners hosted on other sites that you might find appealing.

- Small with Orange Background
- Large Drupal Developers Icon
- Small with Yellow Background
- A Series of Banners based off of the Drupal Head

If you have more, please feel free to submit them to the contributions repository.

Logos for special events

For special events and holidays, the drupal logo is sometimes featured or altered.

The logo was used for the drupal conference 2005 in Belgium and is available here

Booklet

Here you can preview and download a drupal PDF booklet, that can be used for promotion of Drupal. If you want to aquire printed paper versions, you can get in contact with drupal on info@drupal.org to discuss the use, amount and shipping costs.



The booklet in PDF format

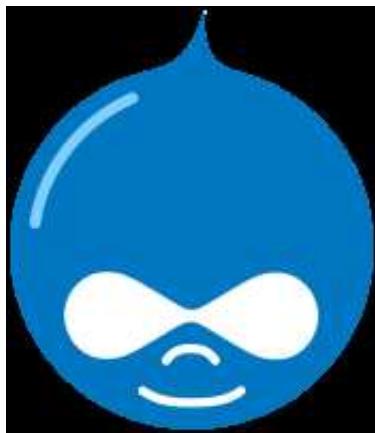


A collage of the booklet

Druplicon

If you wish to use or edit the Druplicon, you can use following logos (all logos use RGB color):

Bitmap versions



PNG version



PNG version 3D celshaded

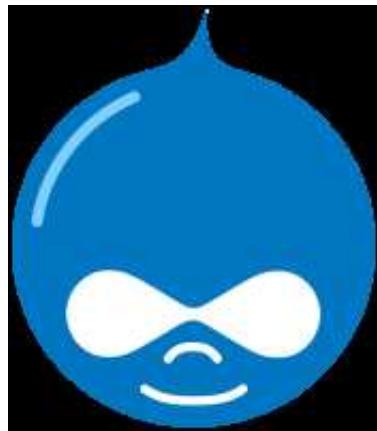


PNG version crystal look

Vector formats



SVG version 3D celshaded



EPS version

Other formats

Paint Shop Pro version (www.jasc.com).

if you can provide other useful formats, please add an issue

Logo colors

Web color

Main drop color: #0077C0

Light-shade color: #81CEFF

RGB color

Main drop color: 00, 119, 192

Light-shade color: 129, 206, 255

CMYK color

CMYK color is used in 4-color printing and prepress industry. There's no 1:1 match in RGB and CMYK color, so it's difficult to bring out corresponding color values. One combination to try is CMYK 91,27,2,0 for main drop color and 47,3,3,0 for highlight.

Presentations

Vancouver Drupal presentation (Open Office Presentation File)

Portuguese translation of the Vancouver Drupal presentation (Open Office Presentation File)

Translation by: Carlisson Galdino.

Drupal - gluing people and code together - Presentation by at Open Source Content Management

The Fosdem 2005 presentations will soon follow. For now, we invite everyone to collect the various bits and pieces of information and list or post it here.

A drupal presentation at an open source fest in Tilburg, the Netherlands. It is a presentation in Dutch, about Open Source communities and how Drupal can help you with your communities.

Future Drupal Presentations can be found in the CVS Presentations Folder.

Reviews

External reviews of Drupal Business Blog Journal Review Very informative and generally positive review by Harold Jarche and Cameron Bales About.com's Review of Drupal 4 of 5 star review. A short review but true and positive. If you read or write a review of Drupal elsewhere, please post the URL and a short description of the review. You might post some information about the site it was posted on (*language, influence, etc*) and your opinion about the review.

Posters

Drupal has some nice looking posters that can be used for promotion of drupal. Feel free to download them in PDF format. And if you are able to modify, translate or improve the poster, feel free to do so and of course give feedback, so that we can add those modifications here.



[Linux Tag poster in EPS format \(default\)](#)

[Linux Tag poster in EPS format \(small\)](#)

[Linux Tag poster in PNG format](#)



[Wing poster in Gimp XCF format](#)

[Wing poster in EPS format](#)

[Wing Poster in PNG format](#)

Drupal t-shirts and other merchandise

Drupal does not have any official merchandise; however since the Druplicon logo is licensed under the GNU GPL some people have taken it upon themselves to make their own.

Profits should be donated back to the Drupal project, Cafe Press does take a significant portion of the price. If you have printable iron on transfers handy consider grabbing a high resolution Druplicon and making a donation.

Basic Druplicon



Powered by Drupal

Designed by Michael Angeles

High resolution versions are available.



Drupal Blogger



Drupal Online Communities



Powered by Drupal

About the handbook

Acknowledgements

Many people have contributed to the Drupal handbooks. Authors of sections are listed in Book contributors, but there have been many other contributions in the form of comments on Handbook sections, Forum contributions, and mailing list postings.

We'd like to acknowledge the contributions of the following drupal.org users:

carlo, kavaXtreme

Commenting on the handbook pages

Please read the following points carefully before submitting your comment to the *Handbook*. If your post falls into one of the categories mentioned here, it will be rejected by one of the editors. We try to keep the handbook clean and up to date. Comments are hard to maintain, often unvalidated, and will confuse readers; thus, the following kinds of comments are discouraged:

- Bug reports, feature requests or language changes: you're in the wrong place. For support use any of the support options.
- Commenting on the fact that something is not documented. This is where you add to the documentation, not where you ask us to add the documentation. Instead, please take the time to create new documentation and add it for the benefit of everyone.
- This is also not the correct place to ask questions (even if you see others have done that before, we are editing the notes slowly but surely). If you need to ask a question, use the forums or the drupal-support list, or see what other support options are available.

If you post a note in any of the categories above, it will be edited or removed.

Just to make the point once more. The notes are being edited and support questions/bug reports/feature request/comments on lack of documentation, are being deleted from them, so if you post a question/bug/feature/complaint, it will be removed. (But once you get an answer/bug solution/function documentation, feel free to come back and add it here!)

Please note that periodically, the developers may go through the notes and incorporate the information in them into the documentation.

Copyright and licensing

All Drupal handbook pages are © copyright 2000-2005 by the individual contributors and can be used in accordance with the Creative Commons License, Attribution-ShareAlike2.0. This copyleft license (very similar to the GPL) allows anyone to copy, modify, and redistribute modifications of all or part of the Drupal handbook as long as

- the license is included with all copies or redistributions.
- the Drupal handbook is attributed as the originating document.

These conditions can be waived only if permission is obtained from the copyright holder(s). By posting comments to the pages in the Drupal handbook, Drupal site members agree that the comments can be revised and/or incorporated wholesale into the Drupal handbook pages under the licensing terms given above.

Contributors to the Drupal handbook are listed on the book contributors page.

Drupal documentation project teams

The following documentation project teams manage various documentation efforts for Drupal. If you'd like to become an active participant in an existing team or have a question regarding a project the team is working on, please contact one of the team coordinators using the contact form linked from their name. If you have a project or question that doesn't fit with one of the defined teams, please contact the drupal-docs mailing list.

Press & Marketing

Andre Molnar (coordinator)
Robin Monks (coordinator)
Liza Sabater
Andrew Hoppin

Best practices

Roland Tanglao (coordinator)
Steven Peck (coordinator)
Kieran Lal

Technology initiatives: The documentation technology initiatives group will focus on technological aspects of Drupal supporting the creation, sharing, management and publishing of documentation.

We will work with the documentation team and with the Drupal development community to improve Drupal's capabilities and usability for this key aspect of Drupal.

Current areas of focus include book module usability improvements, import/export (offline editing), production of documentation in PDF format, versioning, status overviews, notification of changes.

Djun Kim (coordinator)
 Kobus Myburgh (coordinator)
 Gerhard Killesreiter

About Drupal book

Djun Kim (coordinator)
 Bryan Kennedy (coordinator)
 Kobus Myburgh

About the handbook book

This section of the Drupal handbook contains information about Drupal documentation projects, authoring guides for Drupal documentation, and copyright and licensing notices for the Drupal handbook

Anisa (coordinator)
 Charlie Lowe (coordinator)

Press and Marketing

Critical tasks (time sensitive)

June 15th 2005 Deadline:
 Create text for leaflets to be distributed at OSCON.
 See: the project specification
 and also the related forum topic.

June 21st 2005 Deadline:
 Have final graphic design for leaflets created and approved.

General tasks

- Phase 1
- - Continue to identify targets.
 The continued creation of personae / target profiles is a worthwhile

exercise that will aid in later marketing documentation tasks.

- Create a comprehensive list of features.

Features are already fairly well documented throughout the handbook. The goal of this task is to create a point form list of features. A goal of 6-12 features per core or contributed modules is attainable.

- Translate the list of features into a list of benefits

*e.g. XML aggregation = free content for your web site
or taxonomy system = better search engine ranking*

- Match the list of benefits to the needs of identified targets. Many of the benefits may address the needs of more than one target.

- Collect general "FOSS" (free open source software) marketing materials.

There is no need to reinvent this wheel. The benefits of open source are already extensively documented across the web. We need to collect and showcase these benefits within our documentation.

- Phase 2

-

- Write a chapter (collection of pages) for each of our identified targets.

Rewrite or reuse as much of the existing documentation as possible.

- There will be a lot of repetition in the different chapters (the same thing being said again and again). This is not a bad thing.

- Phase 3

-

- Create complementary marketing materials.

e.g. multiple 'canned' print ads, bench mark testing results, case studies.

Some of these already exist. We simply need to find them a home within the handbook.

- Create a section dedicated to comparisons

For example see: Drupal vs. Plone <http://civicspacelabs.org/home/node/12753> and <http://drupal.org/node/13733>

- Create a media kit

- Create and maintain an archive of all media mentions that Drupal receives

- Improve the Drupal site listings to make listing a site and

associated metadata and searching the listings easy and intuitive;
e.g. If speaking to someone about Drupal at a political conference, be able to point that person to a taxonomy that lists all the Drupal sites that are for political candidates in the USA.

- Also See: Restructuring the "About Drupal" section of the handbook

Coordinate with the "About Drupal" documentation team. Djun and Bryan are the key contacts.

Technology initiatives

This is a preliminary list - please feel free to add/delete/change items.

Tasks:

- get consensus on team role, responsibilities
- decide individual roles
- generate list of issues/focus areas
- prioritize according to needs of doc team

Recent updates

| Page | Updated | Approved |
|---|---------------------|----------|
| Drupal conference, Portland, 2005 | 3 hours 9 min ago | yes |
| Drupal's page serving mechanism JV: add ()'s for consistency when naming functions. Removed reference to PEAR as I believe we no longer support that. Changed two references to common.php to common.inc. RE: Removed one link (made HTML comment to that effect) and updated the modules hooks link. Also updated the link to the menu system developer documentation. BM: fixed extraneous italics puregin: marked up all filenames, function names, variables, URLs as code. Rewrote description of site-specific configuration set-up to reflect changes (sites/default/settings.php rather than conf.php). | 3 hours 54 min ago | yes |
| Drupal conference, Antwerp 2005 Modified wording of sentence linking to the meetings. | 4 hours 34 sec ago | yes |
| Drupal developer sprint, Antwerp 2005 Removed Broadband Mechanics party from Saturday. JV Re-organized all cells to link to sub-pages Updated CivicSpace logo. CM | 4 hours 1 min ago | yes |
| Migration tools | 12 hours 1 min ago | no |
| comments about phpBB to sort a page to keep track of information gathered about phpBB import. | 17 hours 14 min ago | no |

| | | |
|---|---------------------|-----|
| WebDAV API for Drupal James Walker is my Mentor | 1 day 1 hour ago | yes |
| Google Summer of Code 2005 please don't change the path alias as Google links to summer_of_code_2005. BM: updated to show that this is underway | 1 day 1 hour ago | yes |
| Configuration Provided a more exciting introduction to Drupal's configuration options. We want to sell Drupal a little bit while we explain it. - moved information about basic settings up from the "Initial configuration" child node -- puregin. - Polished the text a little - clarity, usage, tone... amplified how to get to the settings page. -- puregin | 2 days 7 hours ago | yes |
| detailed use of interwiki | 2 days 10 hours ago | no |
| Listhandler: connect mailing lists to forums - corrected spelling; made spelling of listhandler and mailhandler consistent; added a different heading for the links to admin pages. -- puregin | 2 days 10 hours ago | yes |
| Locale: multi-language support - simplified text -- puregin Kieran added periods to links and tried to make them sentences. Robin-Syntax. | 2 days 10 hours ago | yes |
| Legacy: remapping of old-style URLs - Removed text discussing 'plans' for the module. Re-wrote the text for clarity. -- puregin - Changed title from "Legacy: handler for upgrades" (which isn't really informative) -- puregin | 2 days 10 hours ago | yes |
| Forum: create threaded discussions Richard Eriksson: Added (in HTML comments) instructions for creating forums in 4.6 Kieran-added noun to second half of a clause. Kieran-add particular topic to differentiate from other threaded discussions. Robin-Syntax. | 2 days 11 hours ago | yes |
| Filter: Input formats for user content changed non existant input filter to t Robin: syntax. | 2 days 11 hours ago | yes |
| Drupal: Drupal sites directory server Made it print the documentation included in the module. Kieran-punctuation, spelling, grammar. Robin: syntax. | 2 days 12 hours ago | yes |

| | | |
|--|---------------------|-----|
| Archive: view content by date Kieran-periods, add block admin links, and second paragraph Robin-formatting issues. | 2 days 13 hours ago | yes |
| Insert an image before the promoted nodes on the frontpage | 2 days 13 hours ago | yes |
| How to insert and use the PHP Snippets in your pages - Please adhere to the style guide -- puregin | 2 days 13 hours ago | yes |
| How to use the PHP Snippets with the front_page.module | 2 days 13 hours ago | yes |
| Display a list of (x) node titles of a specific type | 2 days 13 hours ago | yes |
| Trackback: post remotely Kieran-added links | 2 days 13 hours ago | yes |
| Tracker: viewing new and updated content edited for style - cel4145 Punkcut, syntax | 2 days 13 hours ago | yes |
| Flexinode: new content types It was too technical so I completely rewrote it so regular users could understand it. added links. First two need to be reviewed. Kieran | 2 days 13 hours ago | yes |
| Display a list of (x) most recent weblog entries | 2 days 14 hours ago | yes |
| Accounts and roles - moved this content into its own section from an embedded section of the parent -- puregin - added configuration note when adding modules | 2 days 19 hours ago | yes |

| | | |
|---|--------------------|-----|
| Book: collaborative document publishing Kieran-removed node type and replaced it with content type. This could also be called posts. Add link to enable book navigation block. - changed title from "document publishing with the collaborative book" -- puregin - added new paragraphs describing "Export" functionality. Streamlined and made existing text more uniform. -- puregin | 3 days 2 hours ago | yes |
| Blog: a blog for every user Fixed spacing Significantly updated configuration locations Replaced the second sentence with a longer and more generic description of what blogs are and their features. Some re-wording RE: updated information to 4.6 - removed embedded sections to their own nodes -- puregin | 3 days 2 hours ago | yes |
| Aggregator: syndicating content updated the documentation for 4.5 changed location of administration page to administer -> aggregator removed the "Attributes" section of the "add feed" tab removed the section on bundles and replaced it with two sections on categories removed the "blog it" references, though they should probably return (it should appear in the "Using the News Aggregator" section). See revisions for original text. replaced "feed" reference to note about a list of links to the categories the feed item belongs to (this section needs some edits still) reworded some bits made the links section an unordered list cleaned up the spacing added links to FeedDemon and NetNewswire, though held off on links to web-based aggregators like Bloglines and My Yahoo because it might lead to confusion as to what the Drupal aggregator does. - moved embedded subsections into proper child nodes -- puregin Changed confusing they in third sentence because it was confusing as to whether users or admin. | 3 days 2 hours ago | yes |
| Volunteer: organize people to help added statement about RSVP. | 3 days 3 hours ago | yes |

Book contributors

The following users contributed to the Drupal handbook:

- puregin (97 pages)
- Amazon (49 pages)

- Dries (39 pages)
- ax (34 pages)
- BÄr Kessels (25 pages)
- Kjartan (22 pages)
- Dublin Drupaller (22 pages)
- kika (22 pages)
- cel4145 (22 pages)
- kitt (21 pages)
- Steven (19 pages)
- moshe weitzman (18 pages)
- nedjo (15 pages)
- killes@www.drop.org (14 pages)
- bryan kennedy (13 pages)
- drumm (13 pages)
- Boris Mann (11 pages)
- Robert Castelo (11 pages)
- jvandyk (9 pages)
- TDobes (8 pages)
- nysus (8 pages)
- bertboerland@ww... (8 pages)
- sepeck (7 pages)
- robertDouglass (6 pages)
- cherylchase (5 pages)
- al (4 pages)
- rivena (4 pages)
- chx (4 pages)
- Junyor (3 pages)
- pz (3 pages)
- punboy@aaron.pu... (3 pages)
- Jeremy@kerneltr... (3 pages)
- fuzzygroup (3 pages)
- tlk (3 pages)
- dan90@drupal.org (2 pages)
- tatonca (2 pages)
- kbahey (2 pages)
- svemir (2 pages)
- jjeff (1 page)
- fls (1 page)
- mrphp (1 page)
- Kobus (1 page)
- aarre (1 page)
- jeremy (1 page)
- chromatic (1 page)

- bobdoyle (1 page)
- rkendall (1 page)
- levavie (1 page)
- olav (1 page)
- teledyn (1 page)
- matthew (1 page)
- mikeraZ (1 page)
- walkah (1 page)
- handelaar (1 page)
- Richard Eriksson (1 page)
- judah (1 page)
- mathias (1 page)
- felipefonseca (1 page)
- Robin Monks (1 page)
- viksit (1 page)
- njivy (1 page)
- John Downey (1 page)
- wundo (1 page)
- xd (1 page)
- Axel (1 page)
- a3196 (1 page)
- Adagio (1 page)
- marccanter (1 page)
- dlesieur (1 page)
- adrian (1 page)
- Uwe Hermann (1 page)
- factoryjoe (1 page)
- irwin (1 page)
- mabby (1 page)
- stefan nagtegaal (1 page)
- taffy (1 page)
- romca (1 page)
- paster (1 page)
- laura s (1 page)
- marky (1 page)
- gordon (1 page)
- THEMike (1 page)
- gufo1 (1 page)
- andremolnar (1 page)
- Eloquence (1 page)
- jesse_idc@civic... (1 page)
- scottsgreavy (1 page)
- AndreasMangold (1 page)