DRUPAL SPECIAL EDITION

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

SPONSORED BY New Relic®

SPECIAL SUPPLEMENT | WWW.LINUXJOURNAL.COM

THE MANY FLAVORS OF DRUPAL

DRUPAL PROVIDES SOLUTIONS TO REAL BUSINESS PROBLEMS

WHY DRUPAL IS NOT A CMS

BUILD AND DESIGN CUSTOM CONTENT TYPES

CREATE YOUR OWN CUSTOMIZED DISTRIBUTION

OPEN ATRIUM: A FREE AND FLEXIBLE PROJECT MANAGEMENT TOOL

+

TREKK: A Drupal Solution for Universities

CASE STUDY: USENIX.org's Migration to Drupal

And Much, Much More!

CONTENTS SPECIAL DRUPAL ISSUE

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., 2121 Sage Road, Ste. 310, Houston, TX 77056 USA. Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

At Your Service

SUBSCRIPTIONS: *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: **http://www.linuxjournal.com/subs**. E-mail us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Please remember to include your complete name and address when contacting us.

ACCESSING THE DIGITAL ARCHIVE: Your monthly download notifications will have links to the various formats and to the digital archive. To access the digital archive at any time, log in at **http://www.linuxjournal.com/digital**.

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at **http://www.linuxjournal.com/contact** or mail them to *Linux Journal, PO Box 980985,* Houston, TX 77098 USA. Letters may be edited for space and clarity.

WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line: **http://www.linuxjournal.com/author**.

FREE e-NEWSLETTERS: *Linux Journal* editors publish newsletters on both a weekly and monthly basis. Receive late-breaking news, technical tips and tricks, an inside look at upcoming issues and links to in-depth stories featured on **http://www.linuxjournal.com**. Subscribe for free today: **http://www.linuxjournal.com/enewsletters**.

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line: **http://ww.linuxjournal.com/advertising**. Contact us directly for further information: ads@linuxjournal.com or +1 713-344-1956 ext. 2.

LINUX JOURNAL

Subscribe to *Linux Journal* Digital Edition for only **$2.45 an issue.**

ENJOY:

Timely delivery

Off-line reading

Easy navigation

Phrase search and highlighting

Ability to save, clip and share articles

Embedded videos

Android & iOS apps, desktop and e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

Guest Editor	Katherine Druckman katherine@linuxjournal.com
Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors
Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan

Publisher	Carlie Fairchild publisher@linuxjournal.com
Advertising Sales Manager	Rebecca Cassity rebecca@linuxjournal.com
Associate Publisher	Mark Irgang mark@linuxjournal.com
Webmistress	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

***Linux Journal* is published by, and is a registered trade name of, Belltown Media, Inc.**
PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel
Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruizenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

Advertising
E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions
E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 980985, Houston, TX 77098 USA

LINUX is a registered trademark of Linus Torvalds.

drupalize.me

Instant Access to Premium Online Drupal Training

- ✓ *Instant access to hundreds of hours of Drupal training with new videos added every week!*
- ✓ *Learn from industry experts with real world experience building high profile sites*
- ✓ *Learn on the go wherever you are with apps for iOS, Android & Roku*
- ✓ *We also offer group accounts. Give your whole team access at a discounted rate!*

Learn about our latest video releases and offers first by following us on Facebook and Twitter (@drupalizeme)!

Go to http://drupalize.me and get Drupalized today!

```
 *
 * Usage: $(obj).insertAtCursor(text);
 *   obj = a textarea or textfield
 *   text = a string to insert
 */
$.fn.extend({
  insertAtCaret: function(myValue){
    return $(this).each(function(){
      // If target element is hidden, don't do it.
      if ($(this).is(':hidden') || $(this).parents(':hidd
        return;
      }
      if (document.selection) {
        this.fo
        sel = d
        sel.tex
        this.fo
      }
      else if (
        var sta
        var end
        var scr
        this.va
        this.fo
        // High
```

FROM THE EDITOR

KATHERINE DRUCKMAN

Drupal—It's Like Ice Cream, Only Better

As *Linux Journal*'s resident Drupal nerd, I could not be more pleased to bring you this special Drupal issue. Drupal really is everywhere these days, and it's available in more "flavors" than most people in the Open Source community are aware of. So in the interest of spreading awareness about my favorite and ever-growing open-source project, we hope you'll find this special issue both informative and inspiring.

I see a lot of parallels between Drupal and Linux, and not just because "distributions" are such a significant part of Drupal's current landscape. Like Linux and the Linux community, I see both the Drupal software and the Drupal community embracing the fact that no one solution is right for everyone's problem. By maintaining the flexibility of the platform, scratching one's own itch can have tremendous benefits for all. To illustrate this, we've put together an issue full of information about some products that have come to fruition as a direct result of Drupal's flexibility, as well as information on how you can take advantage of the same flexibility to put Drupal to work for your specific use case.

Jeffrey McGuire starts off the issue with some opinions on selling the solutions Drupal can provide rather than selling Drupal itself. In the process, he describes some of the products that illustrate Drupal's potential. Diana Dupuis continues by showing how Drupal is much more than just a CMS, and she explains the hook system that makes the magic happen.

Forest Mars walks through the evolution of Drupal distributions, highlighting the struggles and breakthroughs that have led us to the current set of development tools that make packaging specific configurations possible and portable. He then describes the process of making a distribution, so you can all dive in and get your hands dirty.

One of the most popular Drupal distributions is the friendly and useful Open Atrium (**http://www.openatrium.com**), which is a project management software that, according to the Open Atrium Web site, provides "an intranet in a box with: a blog, a wiki, a calendar, a to-do list, a shoutbox and a dashboard to manage it all". Patrick Settle's tutorial shows off

Open Atrium's best feature—the customization made possible by using open-source software.

Readers who have struggled with their testing and deployment workflow will find some comfort in learning from the struggles of others. Barry Jaspan will help you follow continuous integration best practices with Drupal development, a potentially resource- and time-intensive task that can be made easier.

Continuing with the theme of showing off the many flavors of Drupal, both new and old, be sure to take a look at RedHen CRM, a relative newcomer. Offering a Drupal-native CRM solution, RedHen is worth checking out if you are looking for a highly flexible and customizable CRM experience, and Sean Larkin and Lev Tsypin are on hand to give you a tour.

Busy developers will appreciate Oliver Davies' tips on speeding up development with Drupal distributions. He explains how to automate some of the more repetitive tasks a lot of developers struggle with. Then, Tim Loudon provides an in-depth look at the most intriguing feature of the Trekk distribution, which is aimed at Universities. Because many universities struggle with combining legacy content from multiple sources, the team behind Trekk developed Flatfish, an HTML scraping tool. Tim shows how it all works.

For those who are newer to Drupal, some basic tasks can be very troublesome, but Danny Englander's tutorial on theming custom content types will help a lot of beginners over an initial hurdle. He walks through one of Drupal's best features and shows how to get custom content laid out just right using some handy contributed modules and a little CSS.

For more-advanced developers, Nedjo Rogers has some tips on making Drupal distributions interoperable, which will ensure long-term flexibility and ease of expansion. Janez Urevc demystifies the Drupal cache system and shows how to speed up performance, which is a universal struggle. Jody Hamilton, one of the participating developers, gives an in-depth look at the development behind the popular Open Source community site USENIX.org. And finally, Kojo Idrissa is your guide to the most important and most frequently overlooked part of being a successful Drupaler, contributing to the Drupal community.

I hope after reading this special issue, you'll be as enthusiastic about working with Drupal as I am. As any Drupal user or developer will tell you, Drupal's flexibility allows for almost infinite possibilities, but also nearly infinite amounts of struggle. This issue's aim is to help a lot with the latter and inspire much of the former. Happy Drupaling! ■

Katherine Druckman is an HTML-flinging, PHP-hacking *Linux Journal* Webmistress by day and a refined connoisseur of historic architecture and fine Chinese ceramics by night. She usually can be found surrounded by the charm of aging Texas buildings from the pioneer days or appreciating ceramics of the Song and Qing dynasties. Well, either that or sitting in a comfy chair with a laptop. Yeah, probably the laptop thing. Now in her sixth year heading all things LinuxJournal.com, Katherine's experiences in Web publishing for the open-source audience have strengthened her stance as an impassioned Drupal fangirl.

OPINION

Don't Pitch Drupal—Sell Solutions to Real Business Problems

Drupal is a transformational technology.

JEFFREY A. MCGUIRE

There's a lot we can say about Drupal and why you should choose to use it. It is built and constantly improved by thousands of active developers, and is adopted by hundreds of thousands of users around the world from governments to activists, NGOs to media conglomerates, universities to local schools, big business, technology companies, publishers and beyond. Whether you are new to Drupal, assessing using it for clients or for realizing your own projects, or for a world-changing vision, you should know that there is a more compelling reason: Drupal is changing business and the world.

This has happened before in other open-source projects. The first time I read this quote about Linux from Jim Whitehurst (**http://techcrunch.com/2011/08/17/red-hat-ceo-at-linuxcon-i-have-no-idea-whats-next**), familiar to you as the CEO of Red Hat, I immediately felt that we could say the same thing about Drupal. When you read this now, try substituting the word "Drupal" for the word "Linux":

> Linux is a transformational technology. The technology of Linux empowers advancements and innovations that have nothing to do with the technology of Linux. That is to say, Linux supports the development of new business models, as well as new technologies.

More than 3,000 people took the 2011 State-of-Drupal survey (**http://groups.drupal.org/node/175624**).

When asked to "describe Drupal in one word", the overwhelming favorite was "flexible". This flexibility has given rise to a plethora of innovative products built with Drupal.

What Are Drupal Products?

"Products" are collections of functionality that solve a given set of problems; they are tools that get a job done. A tool that helps your average person—or geek, in our case—get a job done better, cheaper, faster or easier than other options has a chance of succeeding.

Drupal's enormous success is thanks to the fact that it lets us build those tools—call them Web sites, distributions or what have you—better and more efficiently than starting from scratch.

Drupal, my CMS of choice, is itself a product. As Zach Chandler from Stanford Web Services said, Drupal is a tool-building tool (**https://www.acquia.com/resources/podcasts/acquia-podcast-44-drupal-at-stanford-and-beyond-zach-chandler**). Drupal's enormous success is thanks to the fact that it lets us build those tools—call them Web sites, distributions or what have you—better and more efficiently than starting from scratch. When you download and install Drupal, you're already taking advantage of millions of hours of effort that the Drupal community has put into it: coding, bug fixing, improvements of all kinds, user and menu systems, database query engine, dynamic page building—all solved and much more.

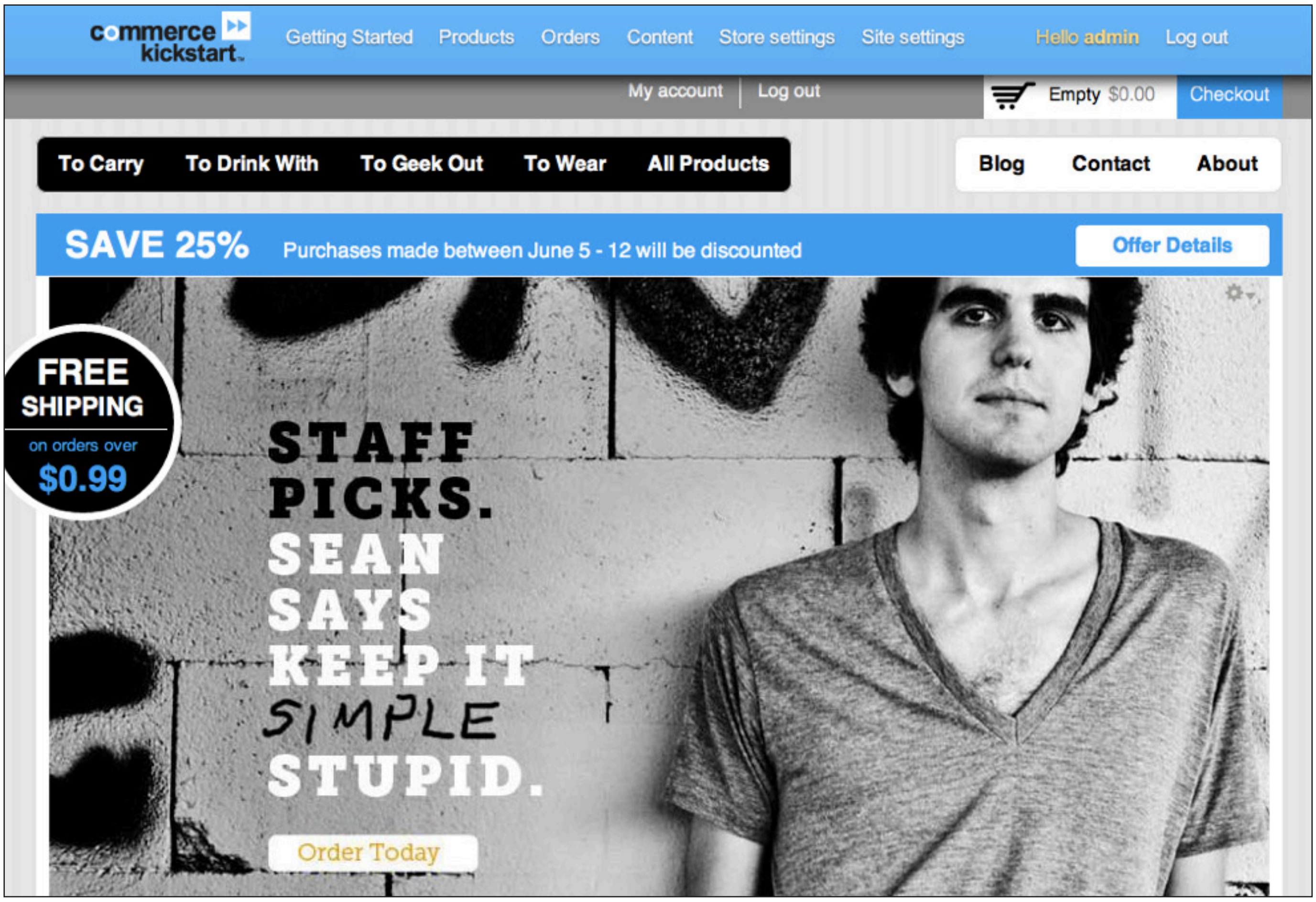
Any of you who have installed plain-vanilla Drupal (**http://drupal.org/download**) know, however, that it still takes quite a lot of work to go from that to a polished, finished Web site: UX and IA planning, configuration of Drupal core, adding and configuring *N* modules, making it all look beautiful in the theme layer and so on.

Drupal products take this a step further. They are solutions to business problems, made by combining the world's best CMS (there, I said it!), with the added value of specialist knowledge and best practices.

For the purposes of this article, I'm talking about Drupal distros—instances of Drupal, preconfigured for a particular purpose—but Drupal plugin modules, themes and "features" (pluggable functionality and configuration sets) can all be products too.

Some Drupal Products

Here is a small selection of Drupal products, covering a fraction of the solutions people are offering based on Drupal today.



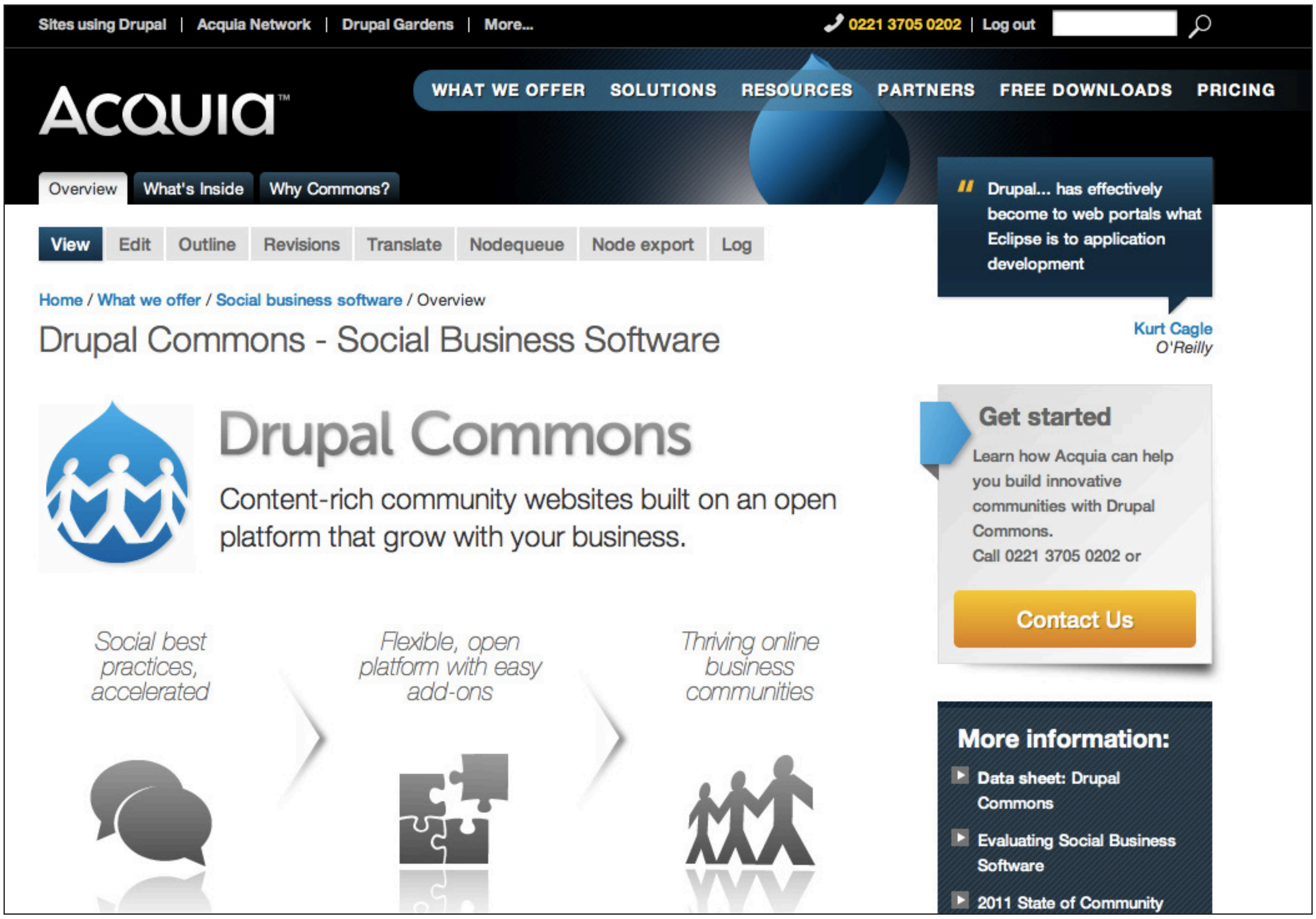
Commerce Kickstart

E-COMMERCE

Commerce Kickstart is built on the Drupal Commerce platform. It's an on-line shop in a box (**http://commerceguys.com/blog/just-released-commerce-kickstart%E2%80%99s-first-beta**), giving you access to the growing e-commerce market while saving you weeks of time configuring the platform.

INTRANET

Open Atruim (**http://openatrium.com**) is an intranet and team portal package. It comes with a blog, wiki, calendar, to-do list, shoutbox and dashboard to manage it all.


Sites using Drupal | Acquia Network | Drupal Gardens | More...
0221 3705 0202 | Log out
ACQUIA
WHAT WE OFFER
SOLUTIONS
RESOURCES
PARTNERS
FREE DOWNLOADS
PRICING
Overview
What's Inside
Why Commons?
View
Edit
Outline
Revisions
Translate
Nodequeue
Node export
Log
Home / What we offer / Social business software / Overview
Social best practices, accelerated
Flexible, open platform with easy add-ons
Thriving online business communities
Contact Us

Drupal Commons - Social Business Software

Drupal Commons

Content-rich community websites built on an open platform that grow with your business.

" Drupal... has effectively become to web portals what Eclipse is to application development

Kurt Cagle
O'Reilly

Get started

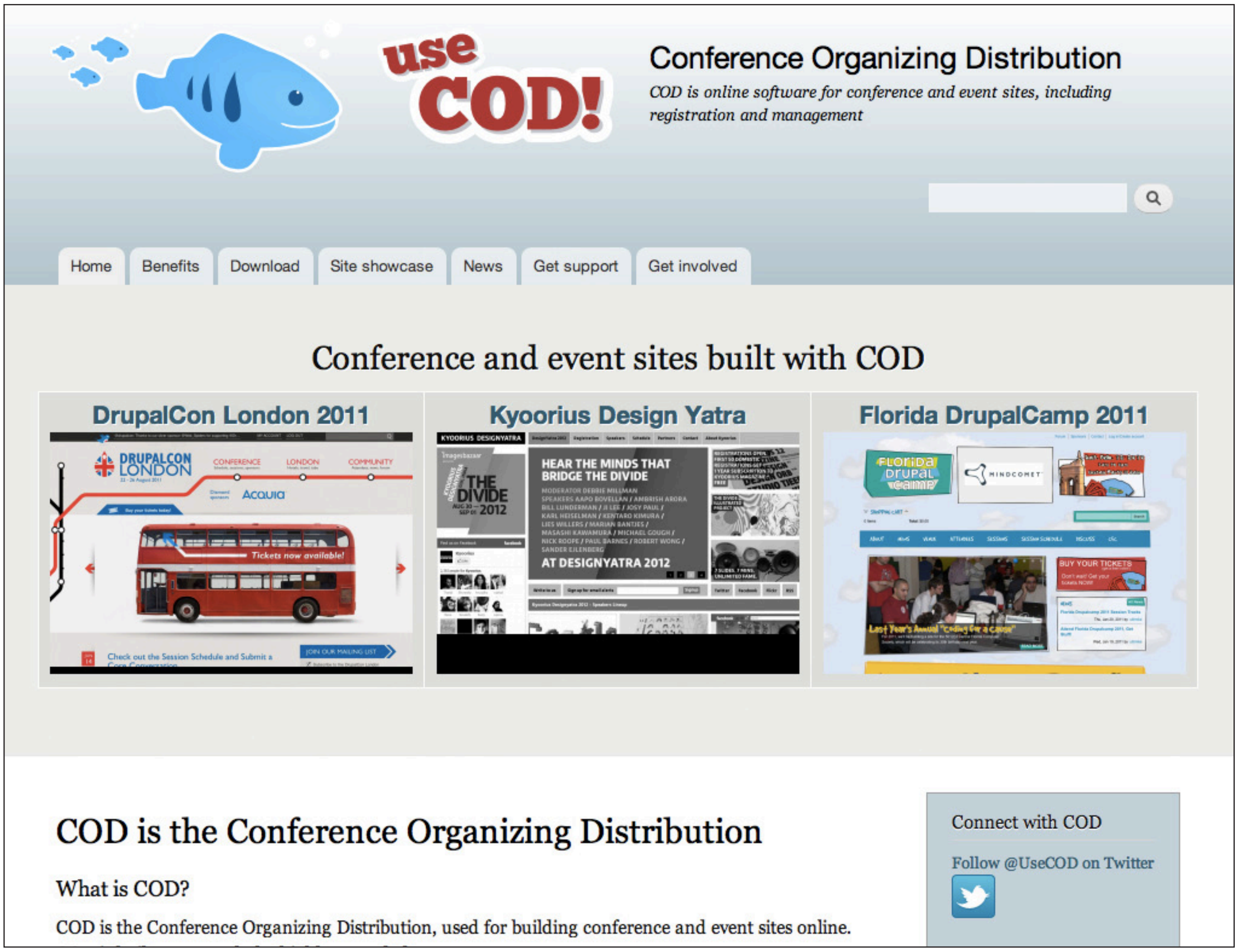
Learn how Acquia can help you build innovative communities with Drupal Commons.
Call 0221 3705 0202 or

More information:
- Data sheet: Drupal Commons
- Evaluating Social Business Software
- 2011 State of Community

Acquia Commons

COLLABORATION

Acquia Commons (**https://www.acquia.com/products-services/drupal-commons-social-business-software**) is a ready-to-use solution for building internal, external and cross-over communities. It provides a complete social business software solution for organizations to add "the social layer in the enterprise".



Conference Organizing Distribution

COD is online software for conference and event sites, including registration and management

Conference and event sites built with COD

COD is the Conference Organizing Distribution

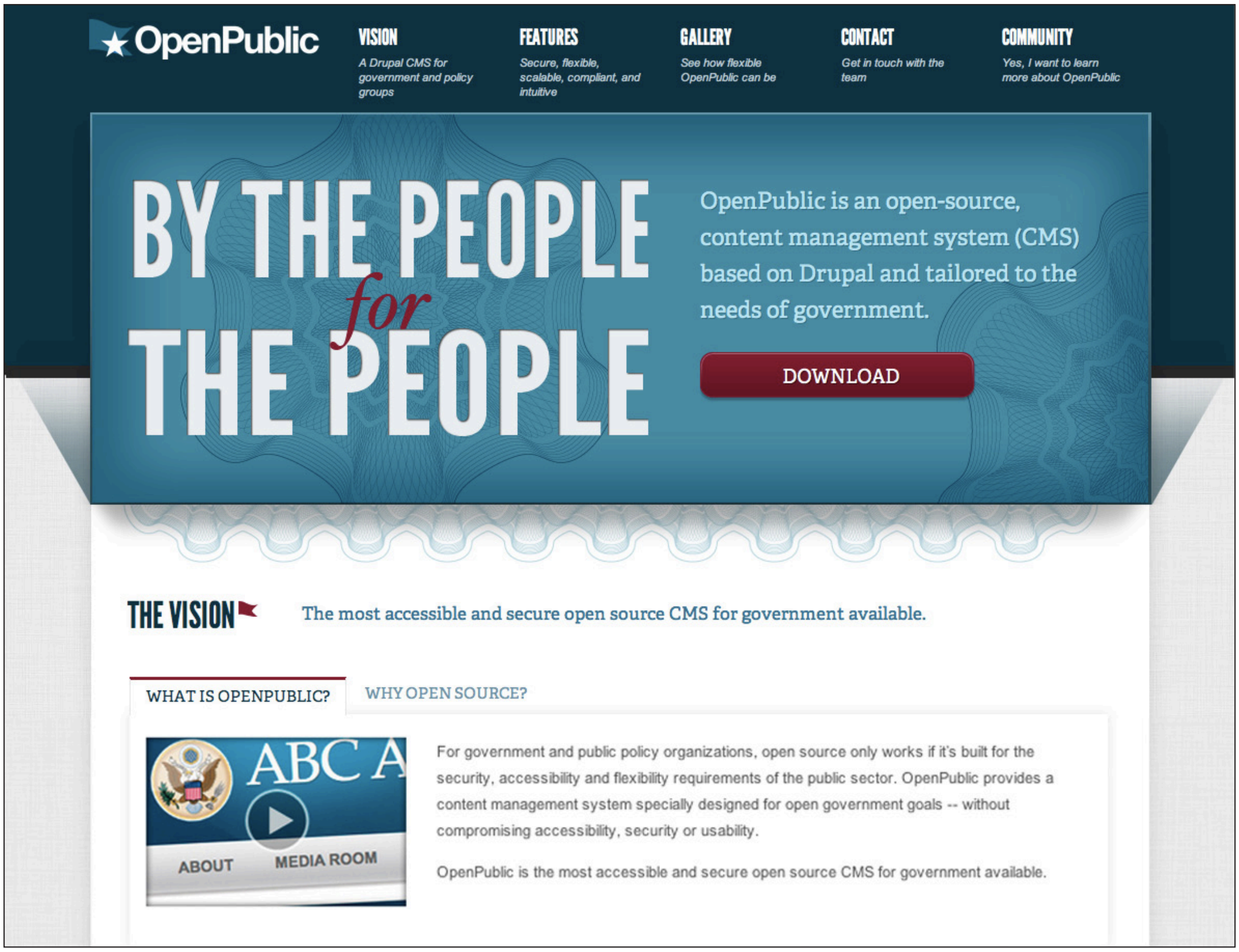
What is COD?

COD is the Conference Organizing Distribution, used for building conference and event sites online.

COD

CONFERENCE AND EVENT ORGANIZING

COD (Conference Organizing Distribution, **http://usecod.com**) is a complete conference and event site that you can customize. It includes e-commerce, session submission and voting tools, multi-track scheduling and more.


OpenPublic
VISION
A Drupal CMS for government and policy groups
FEATURES
Secure, flexible, scalable, compliant, and intuitive
GALLERY
See how flexible OpenPublic can be
CONTACT
Get in touch with the team
COMMUNITY
Yes, I want to learn more about OpenPublic
BY THE PEOPLE for THE PEOPLE
DOWNLOAD
THE VISION
WHAT IS OPENPUBLIC?
WHY OPEN SOURCE?
ABC A
ABOUT
MEDIA ROOM

OpenPublic is an open-source, content management system (CMS) based on Drupal and tailored to the needs of government.

The most accessible and secure open source CMS for government available.

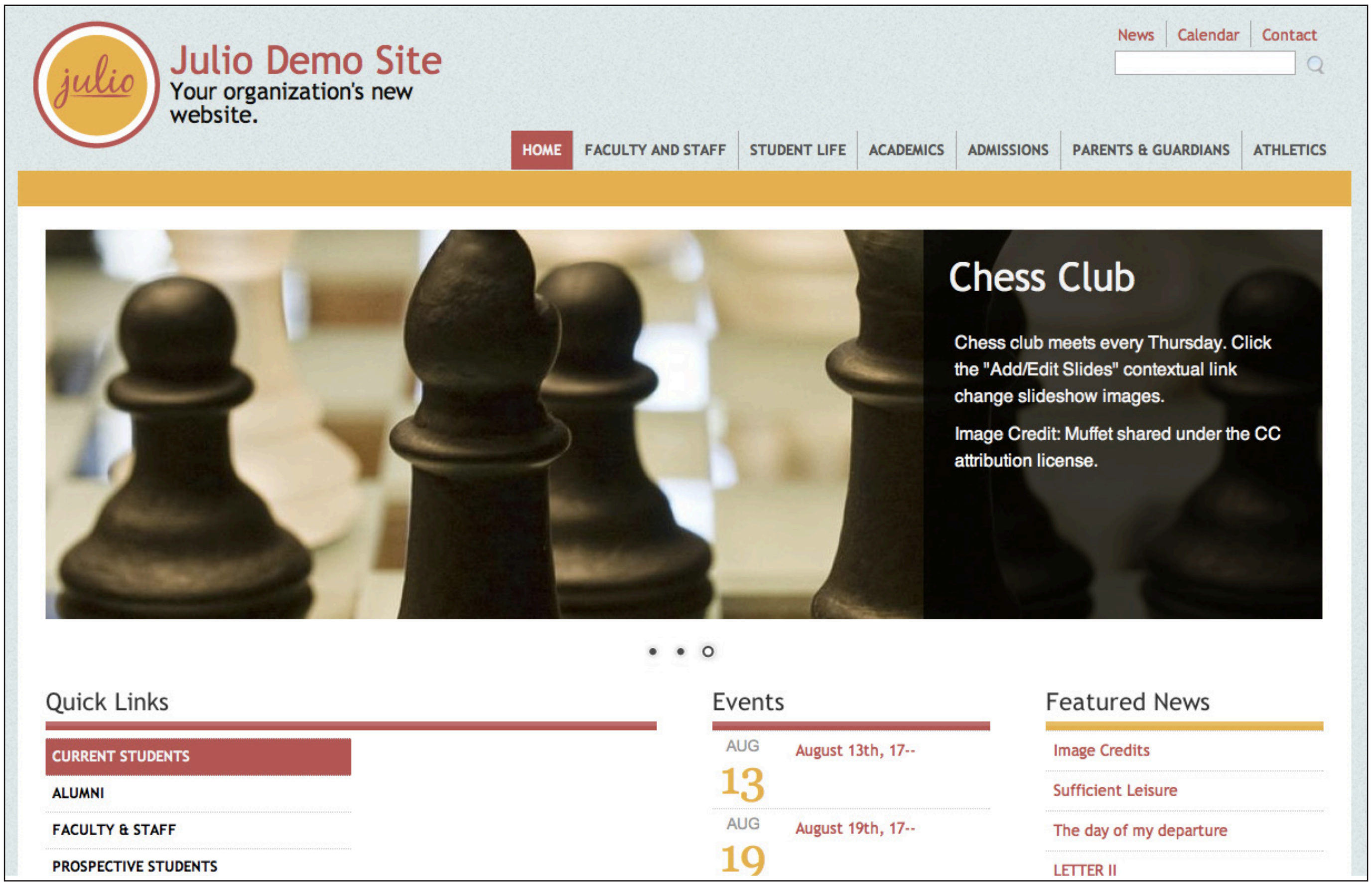
For government and public policy organizations, open source only works if it's built for the security, accessibility and flexibility requirements of the public sector. OpenPublic provides a content management system specially designed for open government goals -- without compromising accessibility, security or usability.

OpenPublic is the most accessible and secure open source CMS for government available.

OpenPublic

GOVERNMENT

OpenPublic (**http://openpublicapp.com**) is a Drupal distro for government and policy groups. Accessibility- and security-compliant out of the box, it focuses on usability, transparency and public participation needed in this day and age.



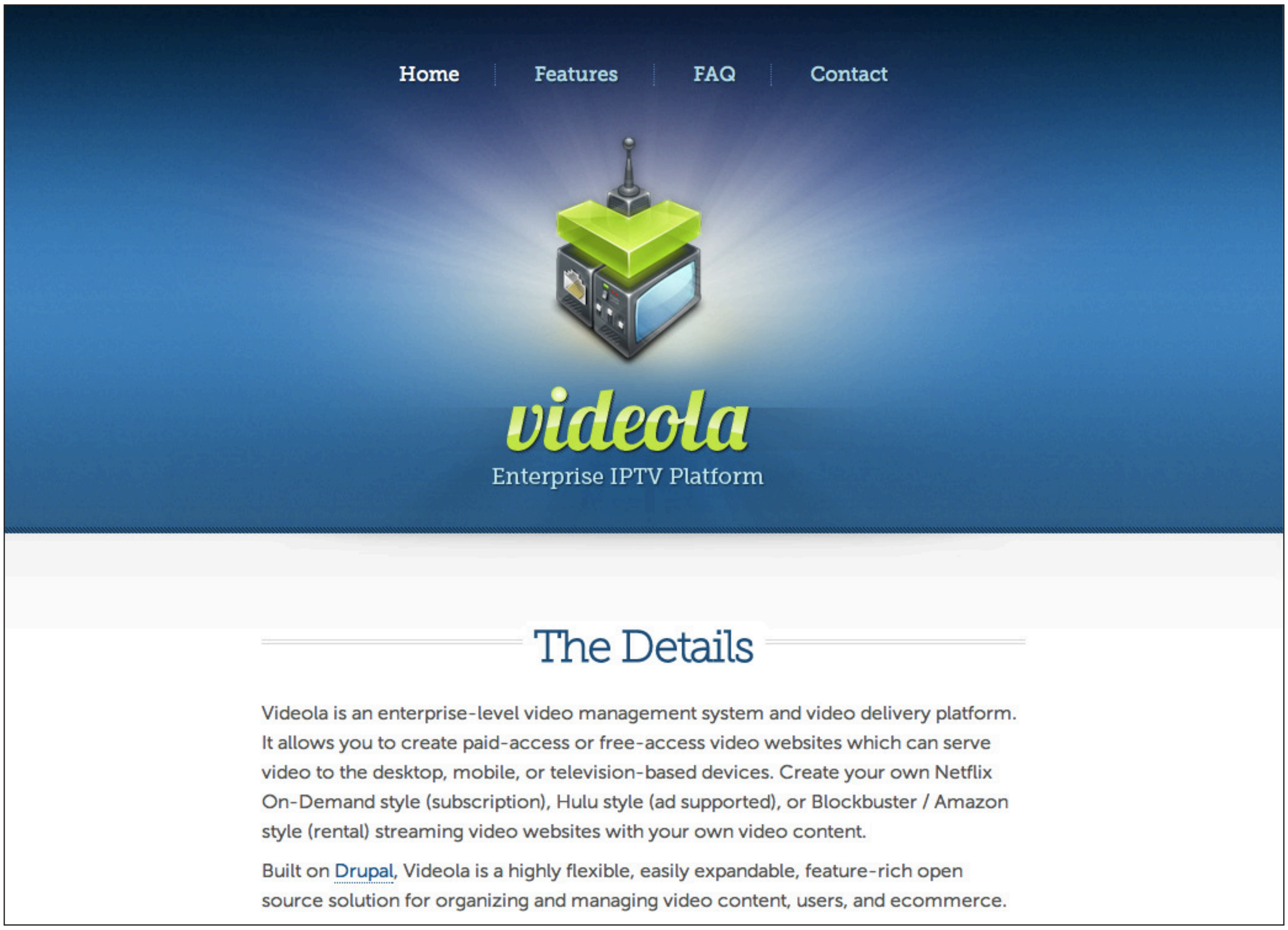
Chess club meets every Thursday. Click the "Add/Edit Slides" contextual link change slideshow images.

Image Credit: Muffet shared under the CC attribution license.

Julio

EDUCATION

Julio (**http://julio.funnymonkey.com**) is a distribution targeted for schools, school districts, small colleges and academic departments within universities. It features calendars, clubs, teams, announcements, departments, staff directories and more.



The Details

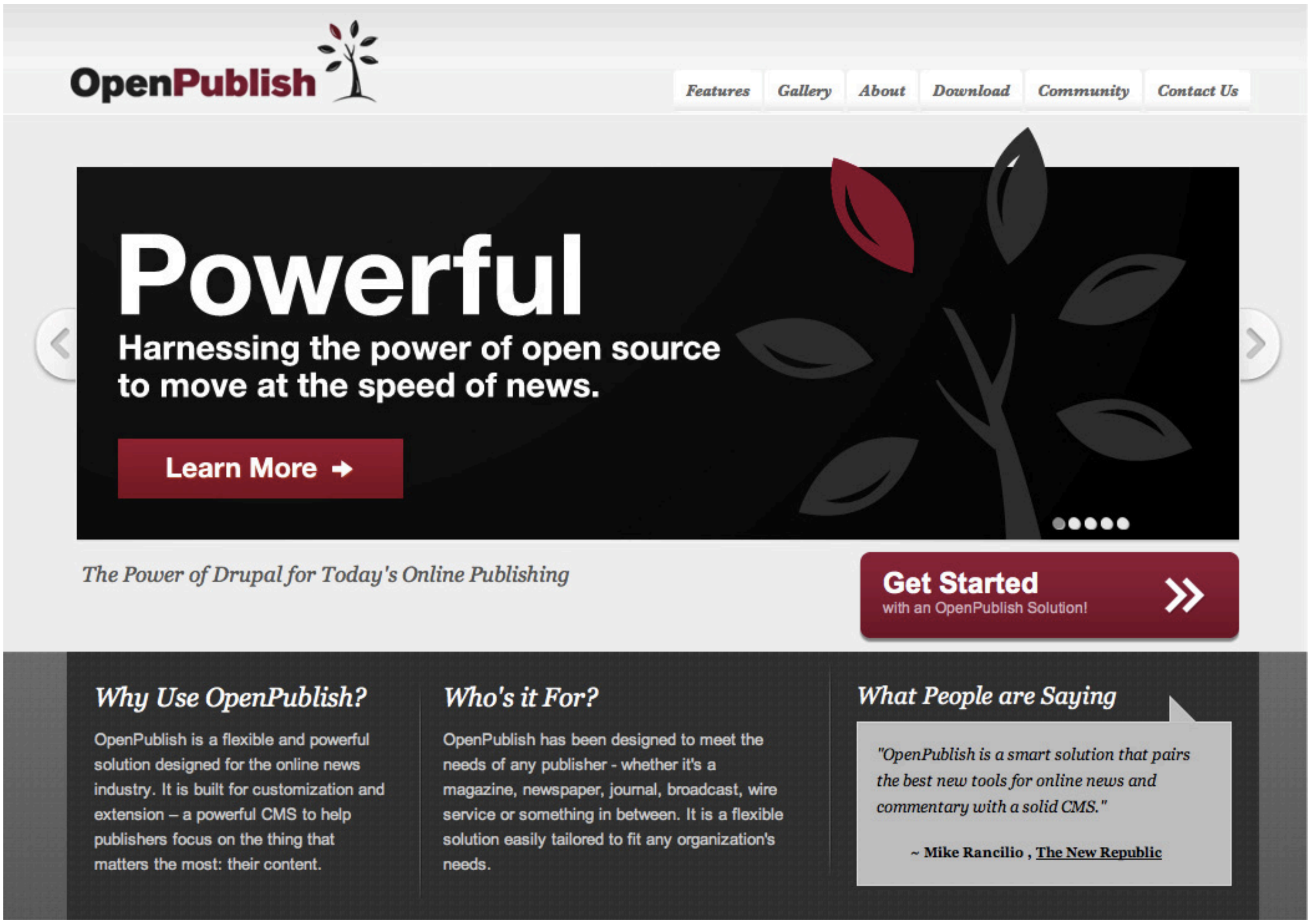
Videola is an enterprise-level video management system and video delivery platform. It allows you to create paid-access or free-access video websites which can serve video to the desktop, mobile, or television-based devices. Create your own Netflix On-Demand style (subscription), Hulu style (ad supported), or Blockbuster / Amazon style (rental) streaming video websites with your own video content.

Built on Drupal, Videola is a highly flexible, easily expandable, feature-rich open source solution for organizing and managing video content, users, and ecommerce.

Videola

MEDIA

Videola (**http://videola.tv**) is an enterprise-level video management system and video delivery platform. Create paid-access or free-access video Web sites with an impressive range of tools for organizing and managing video content, users and e-commerce.



Powerful

Harnessing the power of open source to move at the speed of news.

The Power of Drupal for Today's Online Publishing

Why Use OpenPublish?

OpenPublish is a flexible and powerful solution designed for the online news industry. It is built for customization and extension – a powerful CMS to help publishers focus on the thing that matters the most: their content.

Who's it For?

OpenPublish has been designed to meet the needs of any publisher - whether it's a magazine, newspaper, journal, broadcast, wire service or something in between. It is a flexible solution easily tailored to fit any organization's needs.

What People are Saying

"OpenPublish is a smart solution that pairs the best new tools for online news and commentary with a solid CMS."

~ **Mike Rancilio , The New Republic**

OpenPublish

PUBLISHING

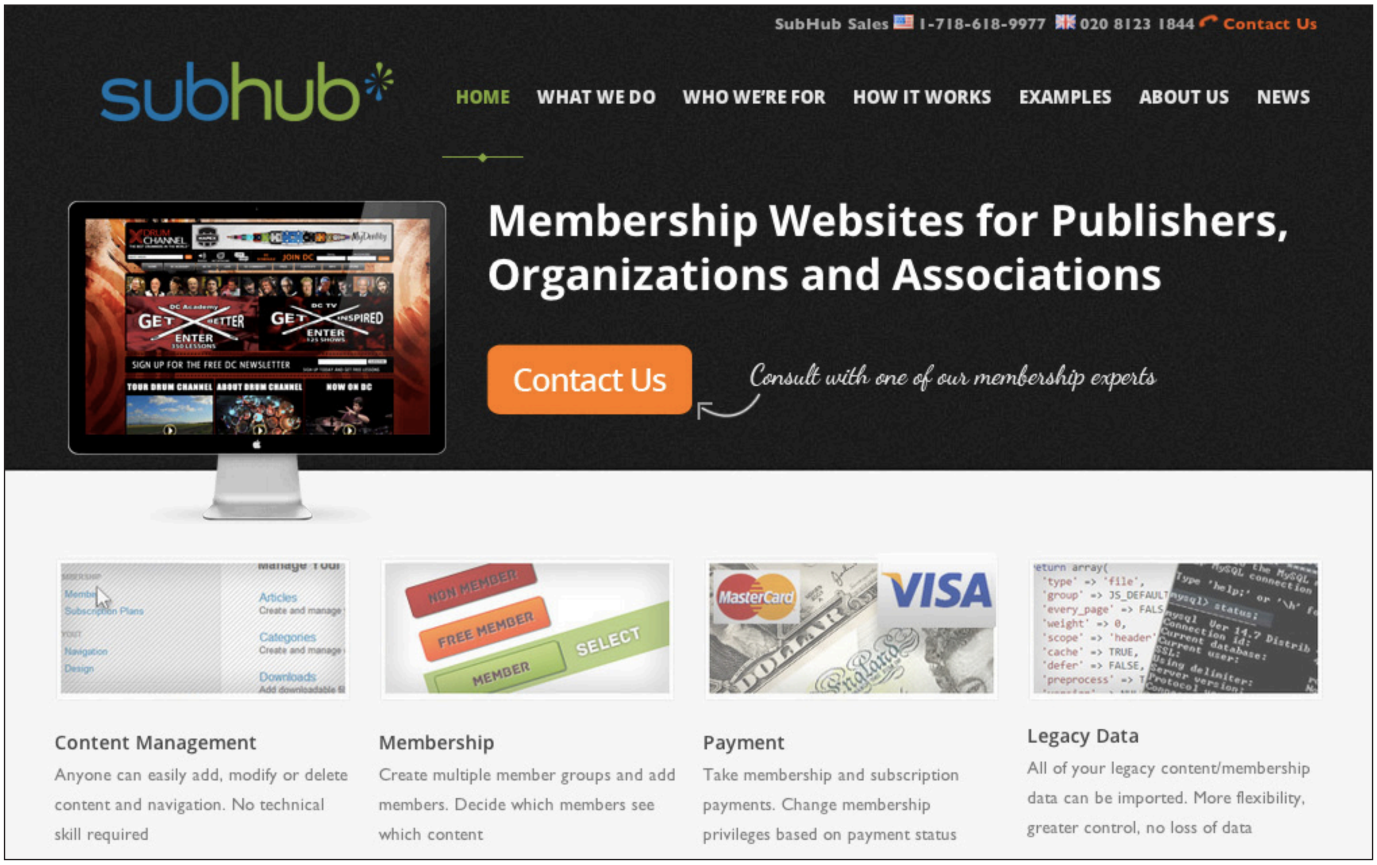
OpenPublish is designed for the on-line news industry (**http://openpublishapp.com**). It contains a powerful suite of content, curation, audience engagement and taxonomy tools to build solutions for today's publishers.

SaaS

There's quite a lot going on in the Drupal Software as a Service (SaaS) area too.

Webform

Webform — one of the most powerful form-building and survey tools on the Web, all built on Drupal, all open source (**http://webform.com**). Pay to get simplicity, speed and service. Or, because it's all based on Drupal GPL code (**http://drupal.org/project/webform**), you can download and set something up yourself as well.


SubHub Sales 1-718-618-9977 020 8123 1844 Contact Us
subhub
HOME
WHAT WE DO
WHO WE'RE FOR
HOW IT WORKS
EXAMPLES
ABOUT US
NEWS
Contact Us
Consult with one of our membership experts

Membership Websites for Publishers, Organizations and Associations

Content Management
Anyone can easily add, modify or delete content and navigation. No technical skill required

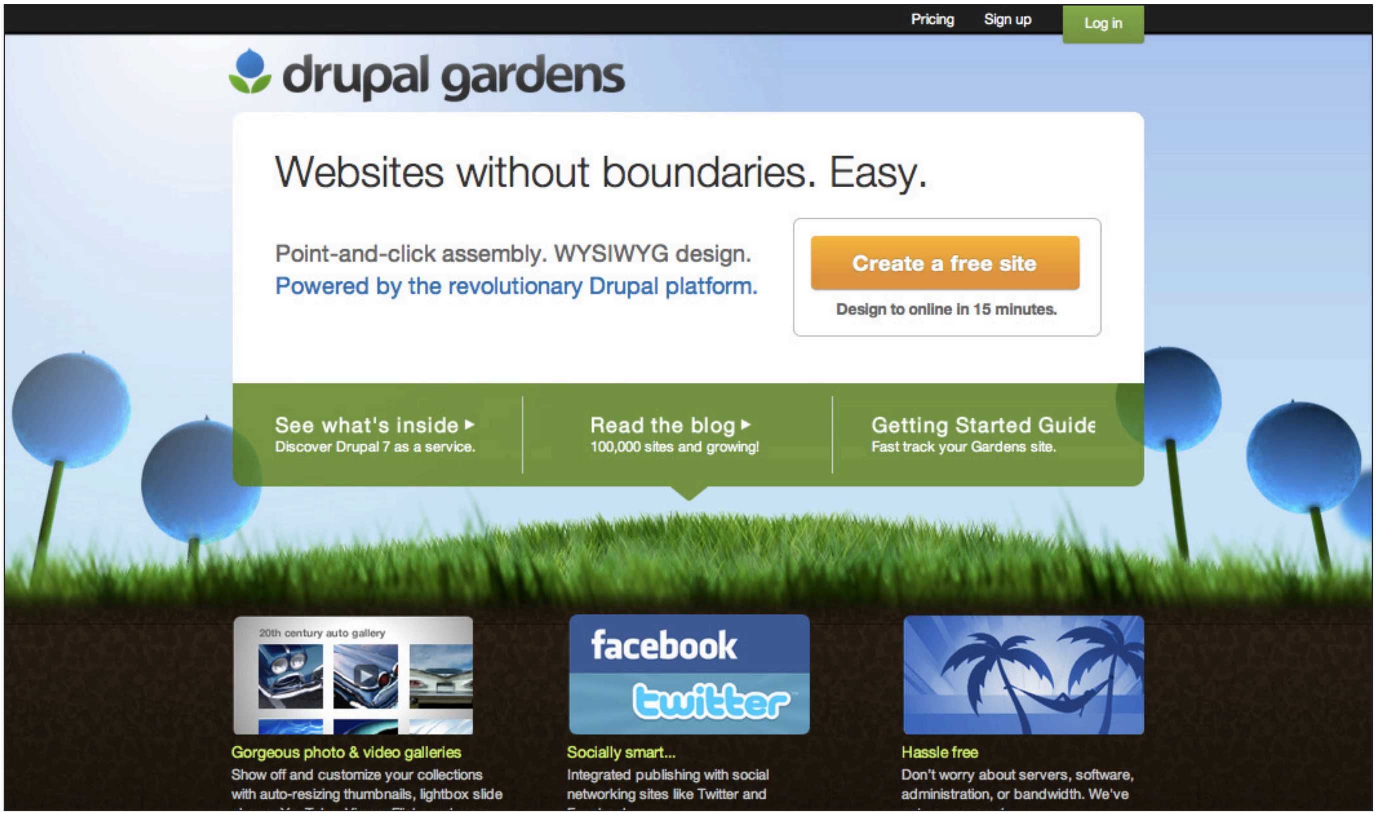
Membership
Create multiple member groups and add members. Decide which members see which content

Payment
Take membership and subscription payments. Change membership privileges based on payment status

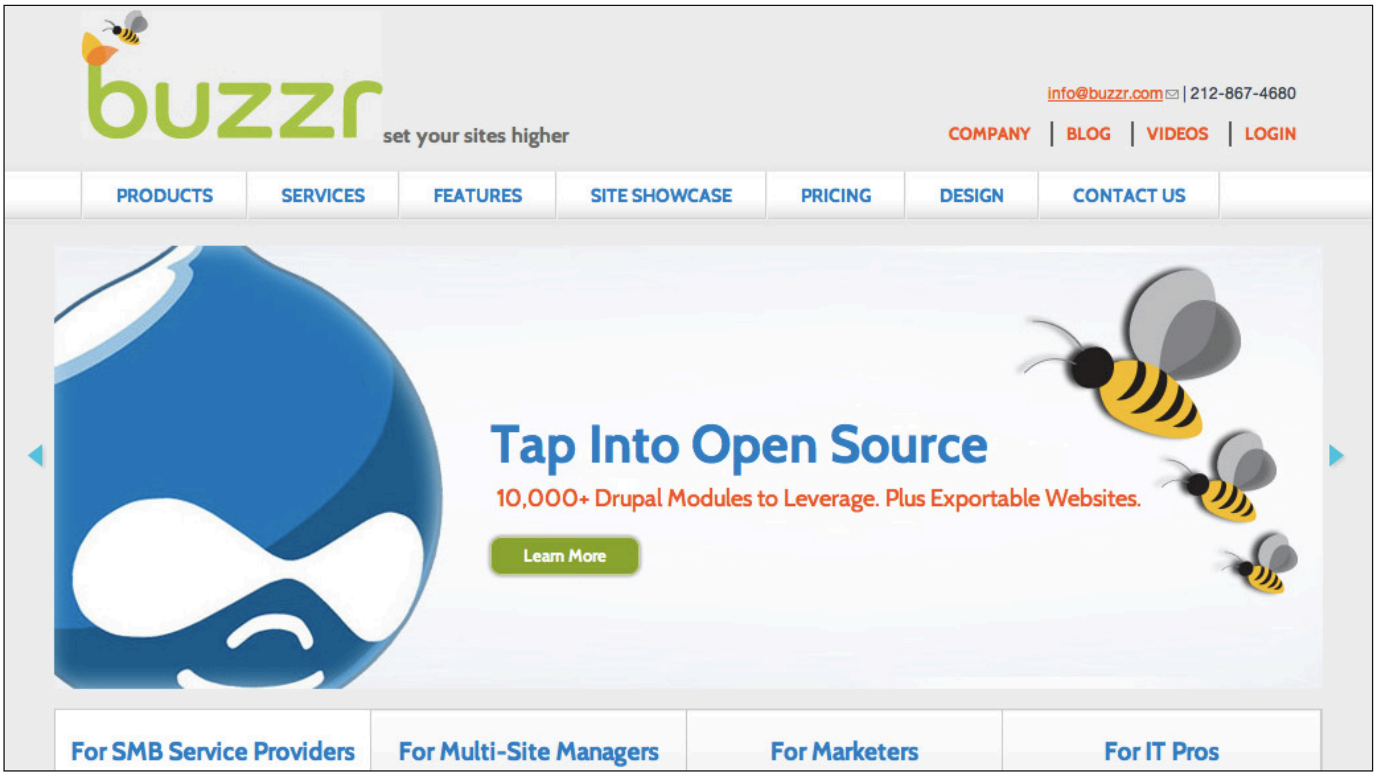
Legacy Data
All of your legacy content/membership data can be imported. More flexibility, greater control, no loss of data

Subhub

subhub, **Drupal Gardens** and **Buzzr** — three different Drupal-in-the-cloud providers, offering a range of services (maintenance, design, support and so on) and options for end users and other service providers (**http://www.subhub.com**, **http://www.drupalgardens.com** and **http://www.buzzr.com**). Drupal Gardens and Buzzr prominently offer no-lock-in guarantees and tools for large stables of sites.


Pricing
Sign up
Log in
drupal gardens
Websites without boundaries. Easy.
Point-and-click assembly. WYSIWYG design.
Powered by the revolutionary Drupal platform.
Create a free site
Design to online in 15 minutes.
See what's inside ▸
Discover Drupal 7 as a service.
Read the blog ▸
100,000 sites and growing!
Getting Started Guide
Fast track your Gardens site.
20th century auto gallery
facebook
twitter
Gorgeous photo & video galleries
Show off and customize your collections
with auto-resizing thumbnails, lightbox slide
Socially smart...
Integrated publishing with social
networking sites like Twitter and
Hassle free
Don't worry about servers, software,
administration, or bandwidth. We've

Drupal Gardens



Buzzr

Six Ways Drupal Products Can Help You

Here are some ways a Drupal product can help. Some of these might apply to you more than others, depending on whether you work with and sell Drupal services, are considering adopting Drupal or want to build your own Drupal product.

Efficiency! Experience! Expertise! I already mentioned that using Drupal gives you a big head start. Now add expert knowledge and experience to the mix. The people who build Drupal products to fulfill the needs of a given segment have gone the extra mile to give you their insight into it and solutions to its problem spaces.

If you need a specialized site, one of the advantages of distributions is that they often are maintained by companies you can hire for support or customization.

If you are building a site based on a specialized distro, you get all that experience at minute zero of your project too. Combined with everything Drupal brings to the table, it all adds up to a huge head start that multiplies your own knowledge, expertise and service offerings.

Customize to your heart's content! On top of all that a Drupal product brings to the table, distributions are customizable, extensible and compatible with all the rest Drupal has to offer. Drupal SaaS offerings are something of an exception to this, but they have other benefits.

Common problems! Until recently, Drupal didn't have an "official" roadmap (the Drupal 8 initiatives, **http://buytaert.net/mobile-for-drupal-8**, are a first for Drupal in this respect), relying on thousands of geeks taking advantage of the "scratch your own itch" principle allowed by the four freedoms (**http://en.wikipedia.org/wiki/Four_Freedoms_(Free_software)#definition**) to add functionality and innovation. I call this the "water rolling downhill" model of development: 99% or more of business problems are non-unique, so the water flows where it is needed. In Drupal's large repository of contributed modules (**http://drupal.org/project/modules**), you are likely to find one or more modules that provide any given functionality you're looking for. If the "contrib" repo is a sunny day, the lenses that focus this potential into fire-starters are Drupal products. They combine, test and refine functionalities into solutions for common problems.

Help with demos! If you offer Drupal site-building services, not all of your prospects have your vision and understanding of Drupal's massive potential and power as a tool-building tool. Not every kid sees the castle, spaceship or city in a giant tub of LEGO bricks. Instead of showing out-of-the-box Drupal and promising "There's a module for that", show them the picture on the LEGO box! Drupal products look like what they do: Julio looks like a school Web site; Drupal Commerce Kickstarter 2 looks like an on-line store. Reduce the amount of effort needed to visualize the final result, and you will reduce the amount of convincing you need to do to turn a prospect into a happy, Drupal-using customer.

Specialize! Hey, you know a lot about... something! Maybe it's hotel management (**http://drupal.org/project/rooms**), real estate (**http://groups.drupal.org/real-estate**) or dentistry. Be the expert. Multiply all the value described above by your own expertise. Build a Drupal product once; reuse it *N* times! This is a great business model followed by many successful Drupal shops and other businesses around the world, whether you sell services based on your distro or simply want to accelerate your site-building business.

Don't pitch Drupal! Finally, Drupal

ACQUIA

Inc. 500 2012 **#1** Software Vendor and **#8** Fastest Growing Overall

Introducing a free tool for Drupal users

Grade your site on performance, security, SEO, and Best Practices.

Instant Insight

Go to **https://acquia.com/instant-insight**, enter your URL, & get a free site scan products allow you to avoid the biggest mistake most Drupal business people make: selling the Drupal value proposition as if others cared about Drupal. You should help them understand the advantages of an open-source CMS that has reached critical mass resulting in innovation, risk mitigation, cost reduction and future-friendliness (**http://www.lukew.com/ff/entry.asp?1407**). But, putting it bluntly, Drupal is just a means to an end, so: *don't pitch Drupal; sell solutions to real business problems.*

Finally, Drupal products allow you to avoid the biggest mistake most Drupal business people make: selling the Drupal value proposition as if others cared about Drupal.

Does a Focus on Solutions Work?

In 2001, one company had the vision to focus on selling solutions: Apple. Given that Apple now has more money in the bank than most countries, we can say it paid off.

Note: much of Apple's business model relies on being closed and proprietary, which goes against the values we live by in the open-source world. The point of this example is that it shows how successful *selling solutions* can be. I am convinced we can be even more successful, combining the collective power and innovation of open source with solutions-based approaches. Red Hat is a great example of this—doing a billion dollars of business in 2011 with an open-source, solutions-based business.

Take a look at this video: **http://www.youtube.com/watch?v=OJtQeMHGrgc**. That was Steve Jobs in the very first Apple store before it opened in 2001. Here's what he said:

> The center half of the store...is devoted to solutions because people don't just want to buy personal computers anymore, they want to know what they can do with them, and we're going to show people exactly that. We've got four sections, and the solutions we've chosen to feature for now are music, movies, photos and kids....You can bring your kids into our store and they can just sit a spell, play their favorite games, and...we have the best selection of Mac education software that I've ever seen, and you can buy the best educational titles for your kids.

Just like Jim Whitehurst's statement about Linux mentioned earlier in this article, this inspired my thoughts about Drupal:

> Our business is devoted to solutions because people don't just want a CMS anymore, they want to know what they can do with it, and we're going to show people exactly that. We've got four sections, the Drupal solutions we've chosen to feature for now are e-commerce, social business, government and education (etc.)....You can show your prospective clients one or more Drupal products, and they can just sit a spell, understanding that Drupal can fulfill their needs.... Today, we have the best selection of Drupal products, distros and features that I've ever seen, and you can implement and customize them to suit your needs.

Don't sell Drupal. Sell solutions to real business problems. ■

Jeffrey A. "jam" McGuire, Acquia Manager of Community Affairs, has a long-standing passion for Drupal and its community. He promotes and advocates for the Drupal project and open-source software. Presenting around the world at Drupal and other events helps satisfy his inner diva, which he also feeds with performances as a storyteller, singer, french horn and alphorn player. Tags: Musician, foodie, Drupalista....

Attend a

DrupalCon

São Paulo - December 6-8, 2012
Sydney - February 6-9, 2013
Portland - May 20 - 24, 2013
Prague - September 2013

association.drupal.org/drupalcon

Attend a

Drupal Business Summit
Expanding the Drupal circle

Dallas ***(09/07)***
San Diego - Higher Ed ***(10/11)***
Atlanta ***(10/26)***
Boston ***(11/15)***

DrupalSummit.com

DRUPAL IS A FRAMEWORK

WHY EVERYONE NEEDS TO UNDERSTAND THIS

Everyone planning and building Web solutions with Drupal benefits from understanding what a "hook" is—and why Drupal is not a CMS.

DIANA MONTALION DUPUIS

One of the greatest challenges that Drupal adopters face, whether they are new site owners or beginning developers, is figuring out what is easy and what is hard to do with Drupal. As a developer, solution architect, technical strategist and even as the friend who knows stuff about Web sites, 60% of my discussions revolve around three questions: how long will it take, how much will it cost, and can my site do [*insert cool new thing*]?

Sometimes, these are easy questions to answer. Many content-related tasks can be accomplished simply by logging in to Drupal, visiting the /admin page and clicking on menu links until you land on the necessary administration page.

More often though, there are complicated questions to answer. Some tasks can be accomplished by adding contributed modules that easily "plug in" to Drupal core, as it comes "out of the box", and expand a site's functionality. Contributed modules are created and shared by the Drupal community and can be added to any Drupal site.

Some tasks require writing custom code, and new modules must be built. Layers of potential functionality are involved in custom features. Some features require communicating back and forth with other sites via an application programming interface (API). Bigger Web sites often require the creation of small applications that accomplish tasks in the background, outside Drupal's usual workflow. In many cases, multiple solutions exist, and choosing one involves giving something up to get something else. As a developer or a stakeholder, finding the best solution that meets business goals and stays in scope depends upon cooperative discussions.

Everyone involved needs to understand that they can architect a Drupal site that offers a more-sophisticated set of features than a WordPress site, because Drupal is not a content management system (CMS); it is a content management framework.

That is where communication often breaks down. Developers are speaking one language while site owners, project and account managers, stakeholders and others involved in the decision-making process speak another language.

When people first learn about Drupal, their initiation often focuses on what a node is, what blocks, content types and views are, and how to create SEO-friendly URLs. These concepts are important, but they frequently fail to answer the essential "how hard is this to do" question or provide a strong foundation for collaborative planning of more complex functionality. Everyone involved needs to understand that they can architect a Drupal site that offers a more-sophisticated set of features than a WordPress site, because Drupal is not a content management system (CMS); it is a content management framework.

Conceptualizing Drupal as a framework does not require years of programming experience; rather, it simply requires understanding what a "hook" is and finding out whether the one you need exists and already is able to do the thing you want done.

To understand hooks, it's necessary to understand how dynamic Web pages, delivered by Web applications, differ from static pages. Most tech-savvy people take this knowledge for granted, especially Linux aficionados and those whose first desktop computer had a flashing cursor at a C: prompt. But many people don't know how Web sites do what they do. (Why would they?) Here is how I explain the difference in layman's terms.

In the olden days, static pages were single text documents containing everything you saw on the page, except for images, in one text file. The file included HTML tags describing the type of content being displayed—for example, <p> denotes a paragraph, and <h1> is a big headline. Browsers (which took ten hours to download) translated this markup and presented pages with a readable structure at a Web address dictated by the filename. The document would be uploaded to the server and saved in the Web site's primary folder. The filename page.html then could be viewed using the browser at yoursitename.com/page.html.

If you wanted to change the Web page's content, you edited that file. If you wanted to change something in the header that appeared on all of the site's pages, you had to edit every page. Whether linking content together or displaying a similar sidebar, content was laid out individually on each and every page by hand.

Nowadays, most sites are dynamic. Small programs, called Web applications, are uploaded and stored on the server. Instead of delivering a static page to view, the program runs when the browser lands on the page, applying logic to the page creation process. This logic dictates how the page is built each time a page is requested (also called "on page load"). For example: the program gets the header, gets the main menu, gets the page's unique content, gets the footer and delivers the whole page to the browser. As a result, now there can be one editable header, one footer and one menu shared among all Web pages.

What about the page's unique content? How does the application "get" that? Imagine a spreadsheet where each row represents each page's unique content. Dynamic Web sites store content in this way. They use a database, which can be imagined as a collection of spreadsheets, called tables. Each table, like a spreadsheet, has columns and rows. Each row has a unique ID. When a page is displayed, the content associated with that page—an article about container gardening, for example—is retrieved from the database table and output to the page.

In Drupal's case, the programming language PHP supplies the logic and MySQL provides the database. Usually, the operating system installed on the server to power this process is Linux, and Apache is the software that handles the requests for pages and delivers them once they are built. This software bundle is called the LAMP stack.

Without static filenames like about.html, how does a dynamic Web site know which row from the content table to display? Drupal, like other Web applications, uses a query string to match the content to the page address. Query strings look like this: ?q=1234, and they are attached to the end of the URL—for example, yoursitename.com/?q=1234. Drupal uses a modified (no less mystifying) address structure: yoursitename.com/node/1234. In both cases, the unique ID, the row number of the page's content, is there: 1234.

Web pages displaying semantic URLs, like yoursitename.com/growing-a-container-garden, have included logic that pairs the unique ID with the words. But for each page, a unique ID still exists and is associated with the content in a database table.

With the advent of dynamic Web applications, the continual development of the programming languages and databases needed to drive them, and the world's voracious need for more and more content-rich sites, *voilà*—the Content Management System (CMS) was born. Drupal is a CMS insofar as it is an application that saves content to a database and displays it to a page using logic that is written into its core or added by programmers. But Drupal is not (really) a CMS; it is a framework that does "CMSey" stuff. Drupal provides the structure for Web applications, far more complex than a CMS, that do all the things Web sites can do: expand the functionality (using contributed or custom code), communicate with other Web applications, run applications written in PHP and other languages behind the scenes, provide responsive pages or integrate front-end languages, scale to handle large traffic numbers by making use of server technologies and provide the foundation for other as-yet-unthought-of innovations.

Here's where the process gets ingenious. But, there is one more conceptual step to take before it's clear that hard or easy depends on hooks—bootstrapping. Again, this is a concept that may seem like common knowledge to the tech-focused reader, but it can be tongue-twisting to explain. Here is my layman's version, which is an oversimplification, but a deeper understanding isn't a prerequisite to understanding hooks.

When a browser hits a Web page, Drupal asks a series of questions. The question process is called bootstrapping. The questions (Q) trigger actions (A).

- **Q:** Who are you (generally) and what do you want? **A:** Initialize and store general info.

- **Q:** Can I just give you a stored copy? **A:** Serve cached data (content stored in memory).

- **Q:** Can I connect to the database? **A:** Do so or die.

- **Q:** Do I need anything from there to work? **A:** Get it.

- **Q:** Who are you (specifically)? **A:** Start a session.

- **Q:** What are your requirements? **A:** Create server/browser page headers (the parameters for further relating).

- **Q:** Where are you? **A:** Select language.

Finally, Drupal delivers the content:

- **Q:** Which page? **A:** Serve up the page.

This is the sweet spot, the place where most (but not all) of the hook magic happens.

Hooks are little blocks of functionality, called functions, that contain PHP code. These blocks of code run when they are called upon. During the bootstrapping process, especially when the final "which page?" question is asked, hooks are called. Whenever an event happens in Drupal, like deleting a page, hooks are called. Inside those hooks, there is code that alters functionality, and it runs as soon as the hook is called. Almost anything you want Drupal to do has a hook doing it.

Drupal relies on naming conventions to call hooks when the time is right for them to run. While building the menu, Drupal looks for hooks with "_menu" in the name. When a page is deleted, hooks with the name "_delete" are called.

Drupal modules override existing hooks or add new ones. For example, if I want to change the way a form is displayed, I put the code for that change inside a function called mymodulename_form_alter. When the form's page is built, Drupal will look for any "_form_alter" function to see if there is more to do. I also can create new hooks in custom code that can be called by other hooks, mymodulename_myhook.

Hooks don't just govern behavior. The theme, which is the collection of files specifically dictating the site's look and feel, also includes hooks. The front end of a Drupal site (the presentation rather than the behavior) is not simply painted on; it relies on hooks as well, all being called when Drupal delivers a page.

Remember our three original questions: how long will this take, how much will that cost, and can my site do [*insert cool new thing*]? The answers, and whether something is easy (quick, cheap and already possible) or hard (time consuming, expensive and innovative), depend on hooks. "I would like my site to do X. Is that easy or hard?" The scale from easy to hard looks like this:

- Drupal already does what you want it to do because the necessary hooks, with the necessary code, run by default.

- Drupal provides an administrative interface for you to turn it on or change it.

- A module or theme already has been written, calling or adding the hooks (with the necessary code inside them) that you need.

- Custom code must be written (using or creating hooks and adding code to them). The time and effort required here varies widely, from three quickly written lines of code to months of programming, creating multiple contributable modules.

- Custom database tables must be created. At this level of complexity, the code still will rely on hooks but begins to run outside of what Drupal does natively; therefore, it is (sometimes) more complex than adding code alone.

- Necessary data comes from other Web sites, or your site's new feature requires communicating with other sites (for example, credit-card processing). The time and effort to do this also varies widely and can be as easy as adding a module (that already handles this communicating) or as hard as writing a separate application that runs when the appropriate hook is called. What your site will do with the data and the load it puts on the system greatly influences the complexity as well.

- Your tasks can't run on page load, a special process has to be written to accomplish them. Sometimes, this is a quick addition (a simple cron job using hook_cron), and sometimes this is complicated. Often, this approach is used when data processing would slow page load down (or take down the site), so it is handled out of sync and saved (cached), serving the cached version when the page loads and the question is asked.

Does Drupal already include the necessary hooks running the necessary code and does it provide an admin interface to set up what you want to accomplish? Easy! Do you need to get mega-amounts of data from elsewhere, process and save it out of sync with page load, and create new database tables that interact with existing data? Hard!

Drupal core and many contributed modules are primarily designed to manage content, to power a CMS—which is why it is right to say, from one point of view, that Drupal is a CMS. Out of the box, users can create any content type imaginable—book reviews, recipes, scholarly paper submissions, press releases, blog posts and so on. An admin interface in Drupal 7 makes creating nodes (the foundational content type with a title and body) and adding fields of related data to them, like the author and publisher in a book review, a code-free task. Creating book reviews that include a cover image, author, publisher, publication date and a link to Powell's City of Books is quick. Adding a five-star rating to each review involves adding one contributed module and turning it on.

How hard it is to make the review look like the design depends on how much the design varies from the way Drupal presents the content to the page. If the author, publisher and so on will be displayed in the order it was created administratively and styled according to the site's general style guide, creating the look and feel involves adding some CSS to the theme's CSS file(s). Easy! But if the page will distribute the fields in a unique order or include custom behavior (like also displaying other books the user has rated), custom work needs to be done. Hooks in the modules and in the theme enable this work to happen, allowing the page load process to be interrupted and edited.

Ironically, the fact that Drupal enables the creation of a book review content type is also what makes it a framework. In the words of Larry Garfield, Drupal core contributor and member of the

Drupal Association Advisory Board:

> What Drupal is today is a tool for building a content management system for a variety of different needs. That's an important distinction for someone looking to build a Drupal site to understand. Drupal is not a CMS. It is the framework with which you build your own CMS, to your specifications, to suit your needs. It is a Content Management Framework.

Diving into the syntax of hooks does require programming knowledge and is, in my experience, where the discussion between developers and product owners should end. My developer cohorts and I discuss the technical aspects of implementing hooks: which to use, where to put them, when to call them, how to simplify the code they run, performance issues and caching plans, the decision to use contributed code or write my own. Once the decision to, for example, pull in feeds and display them is made, the "how" discussions begin. (Node.js anyone?) Communicating the issues that arise as the "how" is being implemented and making collaborative decisions with site owners is the fine art of managing development, the place where the conversations begin again. This process is made easier, every time, when everyone involved understands how Drupal works (framework!) and trusts the easy/hard assessments made by the development team.

Hooks create, define and override the Drupal tools used to build information architecture—associating content with other content and creating navigable structure. Nobody wants a Web site to spit out all of the content in one big blob. The primary tools to build information architecture are content types, menus, blocks, taxonomies and views.

Nobody wants a Web site to spit out all of the content in one big blob.

- Nodes, content types and fields give structure to the content. Fields (like author or publisher) make for easy content creation and visual continuity for the user.

- Menus enable navigation by creating a structure of associations. Menus create a content geography and reveal the paths for exploring it without getting lost.

- Blocks are boxes of content that can be associated with and displayed in a region, such as the sidebar or footer. These boxes can be filled with any kind of content: nodes and content types, menus, lists, text with markup, output like feeds or unique code-created lists like "most recommended". There are hooks, of course, to create, control, edit and override blocks, although most blocks are built administratively.

- Taxonomies are lists of terms that can be associated with content. Most users are familiar with the idea of tagging, associating a blog post (for example) with a list of terms like "coding, biking, cooking or hiking" In Drupal, taxonomies can provide the foundation for more-complex use cases, but associating content is the most common.

- The Views module is a list-maker, powered by a contributed module. Many complex tasks can be handled by Views, but at its most basic, it's the way to create a list of content in Drupal—for example, "all book reviews posted in the last three months". Views also can display content using associations, such as "all posts tagged with the taxonomy terms 'apple' and 'spinach', sorted alphabetically". The lists often are created using the Views administrative interface, but custom code can override the output (hooks!), and entire views can be created in code.

The Drupal framework is a kitchen where, yes, there already are tools in the drawer and ingredients in the pantry. But those tools and ingredients do not define the meals that can be made there. Teams of site owners, stakeholders, project managers, business-goal definers and developers can cook better meals together when Drupal is understood as a framework. Approaching Drupal as a CMS often means bending it to your will: "I want zucchini muffins like my mother used to make; do that." As a framework, Drupal encourages creating the best, most-elegant recipe within the scope of the endeavor: "Here's some zucchini, what can we do with this?"

Drupal's flexibility may make answering our three questions (how long, how much and can it be done) more time consuming. But in the end, the outcome is far more satisfying. ■

Diana Montalion Dupuis is a software developer, Web strategist, writer, trainer and hiker who doesn't spend enough time in the mountains. She lives in Austin, Texas, where it is too hot in the summer, and is Director of Development and Professional Services at Four Kitchens.

DRUPAL DISTRIBUTIONS

Working the Linux Model Up the Stack

The ability to download and run a complete Drupal application easily is being heralded as a new era that is completely transforming how people use Drupal—and the Web. In this article, I discuss how Drupal distributions are made, how you can use them, and how to add your own distribution to drupal.org.

FOREST MARS

In the eight years since *Linux Journal* switched over to Drupal (that fateful Halloween morning in 2004), the Drupal project has undergone explosive growth, maturing from a small but powerful content management system into an enterprise-class development framework powering some of the largest sites on the Web. This process has been driven by a highly collaborative ecology later augmented by the ability to bundle up completed sites and pass them around for use and further development.

In this article, I look at the big picture of this Drupal evolution and where it's going (so that *LJ* readers can better anticipate what's coming) with an emphasis on how the underlying architecture is giving rise to a distribution-based development ecology (in many ways reminiscent of the heyday of Linux distros) and explain how to leverage existing Drupal distributions as well as build your own.

Drupal's evolutionary arc, in many ways, recapitulates the history of Linux development, with the adoption of more automated build tools and package management systems giving rise to a victory garden of different distributions. Drupal distributions are based on installation profiles, which date back to around the time *LJ* switched over its site, and preceded the existence of Drupal shell tools, such as drush and drush make.

Drupal's evolutionary arc, in many ways, recapitulates the history of Linux development, with the adoption of more automated build tools and package management systems giving rise to a victory garden of different distributions.

First, a Little History

Around the time of Drupal 4.5 (the version *Linux Journal* used when it launched its Drupal Web site), the need was recognized for installation profiles as a way to focus "the highly configurable, but largely baffling initial experience" with Drupal into something that could be used and adopted easily by both individuals and groups. Back in the day when the Drupal learning curve was more like a learning brick wall, the idea was to make Drupal make sense to first-time users and to allow it to be used to meet their needs without all the unnecessary complication.

CivicSpace was one of the very first "distributions", although it wasn't necessarily a distribution in the specific sense in which the term is used in Drupal today. Aimed primarily at political campaigns, it was the one that started it all, demonstrating how the Drupal CMS could be used to build an easily expanded, complex distributed network of sites that all communicated with each other. It even included integration with CiviCRM, which was considered pretty advanced for the time.

It wasn't a distribution in the contemporary sense but more correctly an installation profile. Back then, both terms had different meanings from what they have today (as the process has been greatly improved), but all along there has been the goal of facilitating a way to share Drupal site recipes and complete installations easily.

In those days, an installation profile was more like a formula than a completed thing. It included a list of sub-components that would be needed in addition to Drupal core (which wasn't itself even included), such as modules, themes and a profile file that contained scripts required to execute predefined functions and configure the completed site.

Building an installation profile required considerable knowledge of Drupal APIs and a fair amount of time and dedication. In addition to merely having to assemble manually the list of the modules that were needed for the particular installation, you had to write your own custom installation scripts. These scripts invoked PHP functions that could be fired in place of Drupal's default installer and were needed to manage the additional configuration at install time. Since this was before CTools and the Features module allowed you to define exportables, there was no way to prepackage the configuration; you had to do it by writing custom scripts that built

Imagine if you had to manage dependent libraries on a Linux server this way; it's not surprising that it never took off as quickly as some had hoped.

the configuration in real time, essentially "compiling" a working site, but from executable code, not configuration flags.

The final step of using this install profile to generate a finished site—although not involving code writing—was also a bit tedious. The installation profile wasn't a complete distribution, but included a list of everything you would need to get to put it together. You then had to download Drupal core separately, as well as the .profile itself, and then individually download each and every one of the modules. Imagine if you had to manage dependent libraries on a Linux server this way; it's not surprising that it never took off as quickly as some had hoped. Then, you still had to run the installation profile scripts to stitch it all together, and, if all went well, you'd have an up-and-running custom-configured Drupal site.

Enter Drush (Drupal Shell)

The introduction of drush or drupal shell (**http://drupal.org/project/drush**), a command-line shell and scripting tool, completely changed all that and more. At the very least, it made possible far-reaching changes in Drupal development practices. By providing a set of tools that allowed nearly every aspect of Drupal development to be done completely from within a shell environment, it created a foundation for further advancements and techniques—one that borrowed heavily from approaches to Linux development.

Drush is a tool that lives outside the Drupal installation. Installing it on a Linux system is as easy as curling it to the right directory. Drush also can be installed via Debian package management (**http://packages.debian.org/sid/drush**) or by using its custom PEAR channel:

```
$ pear channel-discover pear.drush.org
$ pear install drush/drush
```

Drush can interact with nearly every aspect of the Drupal API and has rapidly grown in popularity, expanding to be able to do more and more using succinct and direct invocations and arguments. Using drush to download every module needed for a given profile meant speeding things up a little, but the giant leap forward was made possible by the release of drush make—essentially a build script that accepts a structured manifest with instructions for "compiling" a specific working Drupal installation. (Of course, it isn't strictly compiled in the technical sense, but the metaphor holds.) This makefile (or "dot makefile") is a text-based configuration file using Drupal's .info format (derived from PHP's .ini format) that's essentially a list of all the required/needed subcomponents. This file describes all the components needed to assemble a full Drupal installation profile, where to get them, how to get them and even what patches need to be applied.

Just a few short months after the first version of drush make was released in mid-2009, it was used to create a Drupal distribution called Managing News, which prior to that had been a downloadable package, but was not based on any universal package management framework. With the adoption of drush, it became the first install profile to include a makefile. This was somewhat of a watershed moment for Drupal, as it was the first Drupal installation profile that automated the build process, instead of requiring you to go fetch all the components yourself. After this, the genie was out of the bottle. Drush make allowed installation profiles to be assembled and served up as a single downloadable tarball, instead of requiring the user to go fetch all the pieces individually from different locations and put them all together.

The next important milestone was Pressflow, which wasn't technically a distribution, or even an profile, but a retooling of Drupal 6 to make its HTTP headers compatible with the Varnish reverse proxy server. Pressflow was important in that it wasn't just a collection of Drupal features aimed at a specific interest group (in fact, it included no bundled modules at all) but was purely a technical distribution that allowed you to use Drupal to do something that wasn't strictly possible with vanilla core: running it on a reverse proxy server to be able to scale site performance. This advancement ushered in a whole new era of highly performant Drupal sites able to serve up content at Web scale, and was later included by default in Drupal 7.

Pressflow also diverged in the sense of being a completely packaged Drupal install containing all components, as opposed to a manifest that you had to build with drush, anticipating the Drupal distribution era. A full distribution doesn't need anything to be fetched over the network but includes all components in a single bundle, vastly simplifying the process of circulating finished Drupal sites with complex sophisticated functionality.

Distributions on Drupal.org

In December 2009, drupal.org updated its infrastructure to integrate the drush make tool as a packaging system (in large part due to the work of Derek Wright). Up until that point, there were no distributions on the main site, but only installation profiles. The integration of drush make turned drupal.org into an on-line package management system that automated the process of turning a Drupal site manifest into a finished Drupal tarball ready for download.

Internationalism blazed the trail, providing one of the first true distributions to be made available there, with the stated goal of making it easy just to install a language-localized version of Drupal with no manual intervention or additional configuration whatsoever. Since then, the Internationalism distro (aka "i10n") has continued to surf the edge; the latest version can be found at **http://drupal.org/project/l10n_install**.

Note: one important distinction between distributions and installation profiles (in the older sense of the term) is that not all distributions can be made directly available on drupal.org, because there is no third-party code hosted there. Those distributions bundling third-party code will have to be content with having only their manifests there and/or making a full tarball (or zip file) available elsewhere.

It's also important to note that the terms "installation profile" and "distribution" have morphed in meaning as the development process has become more defined. Originally, an installation profile was a list of dependencies the user had to retrieve manually with a script or set of scripts that ran ancillary to the default installer script (install.php), which provided a number of places for them to hook into or override the standard behavior. Nowadays, an install profile is thought of as a specifically formatted set of manifests that are executed by the Drupal 7 installer that can optionally invoke ancillary scripts, which may be less necessary for distros that are mainly bundled features.

Install profiles come packaged with distributions in the sense that every distribution has at least one install profile, but an install profile by itself is not a distribution.

Thus, an install profile technically has essentially the same meaning it did originally; however, the way it's used has changed as the packaging system and drupal.org infrastructure has evolved. Install profiles come packaged with distributions in the sense that every distribution has at least one install profile, but an install profile by itself is not a distribution. Drupal 7 is the first version to ship with more than one installation profile from which to choose.

Properly speaking, a distribution is a

full copy of Drupal packaged together with the additional modules, themes and scripts that give you everything you need to produce a complete and complex Drupal Web site in a single downloadable file. It's a ready-made tarball (or zip) that's completely self-contained.

The Distribution Advantage

The now outmoded method of assembling all these components manually was a tedious process and one that required you to know a good bit about Drupal's terminology and file structure, not to mention having to implement all the required function calls yourself. The process was repetitive and predictable, which made it a great target for automation. In addition to the main benefit of automation and ease, distributions can provide:

- Increased development speed.
- A clean way to install your site and be able to hand it off to others.
- The increased ability to compete with commercial "turnkey" solutions.
- An ecology that encourages collaborative development and community involvement.
- A platform to invent new markets heretofore unthought of.

Drupal distributions improve the factory model when you need to able to spin up or deliver Drupal solutions quickly and efficiently, and like the original installation profiles that gave rise to them, they are especially effective at meeting a widespread use case (just as they are much more efficient at meeting it). Essentially, distributions finally realize the vision of install profiles both in terms of vastly improving the Drupal experience and facilitating an ecosystem now heavily populated with hundreds of distros of every stripe and color, such as the Spark distro, which brings in-place editing to WYSIWYG content management; Open Atrium, an extremely popular groupware/intranet (the Whitehouse uses it as an internal collaboration tool); DataPublic, for creating open data portals; a paste-bin distro; a port of Joomla called GLORilla; and the official TedX Web site, just to mention a few of the hundreds of freely available distros, from educational to "corporative", from scientific to religious, all of which can be downloaded directly from **http://drupal.org/project/distributions**.

Note, however, that Drupal doesn't have a defined way to separate upstream and local configuration and settings. There's no "/usr/local", and apart from workflow-based solutions, it's a missing piece that arguably would be of great benefit to making Drupal distributions even better.

What's Included in a Drupal Distribution?

Distributions are essentially finished installation profiles. The basic components of a distribution are Drupal core plus any of the following additional components:

- Installation profiles.
- Contributed modules and themes.
- Distro-specific features, modules and themes.

Just to be clear, a distribution is an install profile, packaged with Drupal core and any required contrib modules. The install profile that comes with a distribution may in fact have additional scripts that run at installation time in its .profile file. Although distributions and installation profiles seem very straightforward once you understand how they work, you may find the terminology seemingly inconsistent, both in the official documentation and elsewhere. Building your own Drupal distribution, however, and showcasing it on Drupal, isn't too complicated. The following is a short guide on how to roll your own drupal distro and submit it.

How to Make Your Own Drupal Distribution

The first step is, of course, to make an awesome Drupal site that other people would be interested in downloading and using. You can use an existing distribution as your starting point, building on the work of others. For example, Panopoly (**http://drupal.org/project/panopoly**) is an Apps-enabled distribution of Drupal designed to be both a general foundation for site building and a base framework upon which to build other Drupal distributions. After you have a completed site, you're ready to package it up for sharing. The site from which you're making a distribution could be a clean install of the architecture you've built (meaning it could have no created content yet), or it could be a full site with content that's been created by multiple users. The distribution build process will pack up only the modules, not any of the configuration, content or users. To include configuration, you'll need to write a .profile with bundled scripts (or capture it all in Features). To provide default content or even users "out of the box", you'll have to use the Profiler module (**http://drupal.org/project/profiler**).

The second step is to generate the makefile. This file is a manifest specifying the version of Drupal core that is needed to build the site (if none is given, the latest stable release will be used) along with a list of all contributed and custom modules. Contrib modules optionally may be given a version number, and custom modules generally will need to have a path to a repository to download them from, such as GitHub. (You also can use a repository other than drupal.org for contrib modules, such as running your own local repository to speed up package retrieval.) The easiest way to generate the basic site manifest is to use:

```
$ drush make-generate my-site-name.make
```

which will auto-generate the drush makefile based on your current site configuration. For any information it can't find (the path to a custom repository, for example, which isn't stored anywhere else in configuration), it will insert placeholders reminding you to fill this in manually. And, of course, if you're using alternate or dev versions of any contrib modules, they will be included too. Inspect the file it created for you and make sure any placeholders are replaced with corrected paths. (Covering the exact syntax for each of these entries is outside the scope of this article.)

Note that drush make handles only modules and libraries (downloadable code) and not the configuration stored in the database (including content types and so on) for which you'll need a manually built install .profile or bundled "features" (using the Features module).

The next step is to execute your makefile to prepare your distribution using the prepare-install command:

```
$ drush make --prepare-install --tar my-site-name.make my-site-name
```

(Note: this step can take several minutes depending on the size of your site.)

At the completion of this step, you will have a finished Drupal distribution that anyone can use to spin up an instance of the kind of Drupal site you've built, without downloading anything additional over the Internet. You could offer it for download at the site of your choosing, or you could put it on a site specifically dedicated to showing off your distribution and making it available for download.

You even could add your custom distro to the growing number available for download directly from the drupal.org Web site. However, to do this, you won't submit a full package made with the `prepare-install` command, but instead generate the makefile only:

```
$ drush make --generate-makefile drupal-org.make
```

(Note: at the time of this writing, you do have to make one manual edit to this file. See **http://drupal.org/node/642116** for instructions.)

Then, verify that your makefile is ready using:

```
$ drush verify-makefile
```

Finally, you'll want to execute the makefile locally (just to confirm it's working) and then commit the manifest only (that is, the drush makefile, not the fully packaged distribution) to the distribution's repository using Git, the version control system used on drupal.org (and created to manage the Linux kernel). To release a distro on drupal.org, you will need the ability to create full projects there, which involves a one-time approval process. This approval is not instantaneous, so if you don't have it, don't plan for an "immediate" release of your distro.

After you've successfully created and tested your distribution, and uploaded the make manifest, the drupal.org packaging scripts will handle all the rest for you.

For full details on submitting your distro to drupal.org, see **http://drupal.org/node/642116**. ■

Forest Mars is a hypermedia architect probably best known as the founder of New York's first and only free Internet café and the city's first free software-based wireless Internet service provider, as well as "Yellow Hat", the Tibetan language Linux distribution. Recent Drupal projects include architecting a video delivery platform for the world's largest television network and New York City's first civic engagement platform for the borough of Manhattan. He is currently CTO of a successful startup and president of the Community Free Software Group, an established 501c3 foundation dedicated to advancing the cause of software freedom.

aten
design group

Drupal, done beautifully.

We are passionate designers and engineers, dedicated to creating meaningful user experiences for projects that matter. For more than ten years, we've helped journalists, educators, human rights activists, climate change adaptation experts, health care providers, and disaster response teams leverage open source technology to impact lives.

How can we help with your project? Find out at atendesigngroup.com.

atendesigngroup.com • 3507 Ringsby Court, Denver, Colorado, 80216 • 303-831-0448

MAKING OPEN ATRIUM YOURS

Open Atrium provides organizations and individuals with a free, robust and flexible project management tool—one that demonstrates just how powerful and beautiful Drupal can be.

PATRICK SETTLE

OPEN ATRIUM is packed full of features right out of the box: a calendar, discussions, a notebook for project documentation and a flexible task manager. Even with hundreds of thousands of Open Atrium installs, its most powerful feature is often never taken advantage of: Open Atrium is an open-source project.

Although people know what open source means, they don't take advantage of this feature enough, even though it is one of the best features any project could have. This feature empowers you to do so much more, but it doesn't mean that you need to be a hard-core developer to participate.

Let's explore some very real examples of the benefits of Open Atrium's open-source feature and look at how you can take advantage of this underutilized facet to make Open Atrium your own.

Set Yourself Apart

You probably wouldn't install Drupal for your business Web site and leave the default theme. Why would you do that with Open Atrium? Although Open Atrium's Ginkgo theme is a very usable theme, why not let your design and style shine through to your staff, customers and vendors by customizing the look and feel of Open Atrium with your own sub-theme?

One powerful aspect of Drupal's theme system is its ability to use sub-themes. Sub-themes allow you to use an existing theme as a jumping point, saving a significant amount of time, while still letting you apply your own design.

Crystal—a Sub-theme for Open Atrium

A good example of a sub-theme for Open Atrium is the Crystal sub-theme by Daniel O'Prey. Building off a previous sub-theme, Daniel created Crystal to be a clean app-like theme. Although it could be used on its own, let's use it here as a starting point for a new theme.

Installing a Sub-theme

Start by making a copy and renaming Crystal's theme folder into /sites/all/themes. For this example, name the sub-theme Baara, so its folder will be /sites/all/themes/baara. (If you want to use Crystal as is, there's no need to rename anything.)

Like any Drupal theme, this new sub-theme Baara will need a .info file that will define the name of the theme, a description, any CSS style files and regions or DesignKit options. You simply can rename crystal.info to baara.info, and using your favorite text editor, make changes as needed.

When finished, your .info file should look like this:

```
name = "Baara"
description = "A Ginkgo sub-theme to get work done,
➥based on Crystal for Open Atrium."
core = "6.x"
atrium = "1.x"
engine = "phptemplate"
base theme = "ginkgo"

stylesheets[screen][] = "style.css"

regions[left] = "Left sidebar"
regions[right] = "Right sidebar"
regions[content] = "Content"
regions[header] = "Header"
regions[space_tools] = "Space tools"
regions[page_tools] = "Page tools"
regions[palette] = "Palette"

features[] = ""

; Designkit
designkit[color][background][title] = "Background"
designkit[color][background][description] = "Color for
```

```
➥headers, other fills."
designkit[color][background][default] = "#44aa55"
designkit[image][logo][title] = "Logo"
designkit[image][logo][description] = "Header logo."
designkit[image][logo][imagecache] = "imagecache_scale:300x40"
```

The first six lines set the name and description of the new sub-theme, the version requirements for Drupal, and most important, it lets Open Atrium know that it's a compatible theme and defines the parent theme for the sub-theme (in this case, Ginkgo).

Sub-themes function just like normal Drupal themes. They can include images, stylesheets and templates. Stylesheets can be reused from the parent theme, or in this example case, be overridden by declaring them in the .info file, as seen on the line `stylesheets[screen][] = "style.css"`, which will override the style.css file found in Ginkgo.

The "regions" entries list all regions available in your sub-theme. If you modify any of the template files to add new regions, you'll need to be sure to modify this list. It's important to note that while adding regions can be done easily, removing or renaming regions should be done with care and may cause complications, as many Open Atrium Features are built to expect those default regions.

If you plan on having additional customization that could be managed through the Drupal admin pages, the "designkit" entries allow you to specify the customizations you want to make available on /admin/build/themes/settings/baara. To learn more about DesignKit and how to use it, see the README.txt, which comes with the DesignKit module (**http://drupal.org/project/designkit**).

For additional information on Drupal theming and details on available options to use in the baara.info file, read the on-line Drupal Theming Guide (**http://drupal.org/documentation/theme**).

Ginkgo Sprite Overrides

With the new sub-theme folder created and .info file completed, you can turn your attention to applying style and design. Ginkgo's extensive use of sprites throughout its design allows you to make some pretty radical changes simply by modifying or replacing these images.

Ginkgo uses these sprites as background images to many of the elements throughout the site as defined in the style.css stylesheet. There are three main sprite images (along with RTL for right-to-left languages), which you will find in the Ginkgo images directory:

- sprite_base.png — base UI elements, such as priority indicators and default user avatar placeholder images.

- sprite_icons.png — Atrium Feature's icons used in menus.

- sprite_skin.png — gradients and other backgrounds, along with the default logo.

To override these images in your sub-theme, re-declare them in your style.css and include the modified sprite images in your Baara theme's images folder. If you want to replace the default icons, you can modify the sprite_icons.png by inserting the custom icons. More-advanced sub-themes may take advantage of additional sprites, which you can include in your sub-theme's image folder.

Be sure to maintain the same metrics and coordinates of these sprite files from the originals; third-party Open Atrium features are built with this expectation.

A Style of Your Own

In the example .info file, you made the declaration `stylesheets[screen][] = "style.css"`, which overrides the Ginkgo theme's style.css stylesheet. This not only allows you to override Ginkgo's sprite images, but also will house any other CSS style changes you need.

Be careful when overriding any of Ginkgo's design, as changes to templates and layouts may make it difficult to take advantage of community-contributed features.

If you don't have the time to create your own sub-theme, check out **http://community.openatrium.com/shopresources/feature-directory** for a listing of sub-themes created by other Open Atrium community members.

When Something Is Not Quite Right

Now that Open Atrium is looking more like your organization, let's consider the actual features of Open Atrium. Like a great many things in life, one size does not always fit all, and each organization that uses Open Atrium often has unique needs and requirements that require changes to the default settings included in Open Atrium.

Overriding Settings

There are three ways to override the settings in Open Atrium: simply make the changes in the Drupal configuration interface, fork the Atrium feature you want to modify, or create the much more flexible custom feature override module.

The first option—making changes directly through the configuration pages—is immediate. It can be done by almost anyone with access, and it requires no programming. Although this sounds great, note that this technique comes with a great deal of risk—it's

Table 1. Some Common Drupal Customizations

COMPONENT	CUSTOMIZATION	SOLUTION
Blocks or boxes	Show or hide	Modify the context that sets the block as visible or hidden (see the section on contexts).
Boxes	Modify output	Use hook_boxes_view_alter(&$box, $delta).
Contexts	Modify settings	Use hook_context_default_contexts_alter($contexts) in a custom module.
User permissions	Modify settings	Use hook_user_default_permissions_alter(&$permissions).
Imagecache presets	Modify a preset	Use hook_imagecache_default_presets(&$default_presets) in a custom module.
Menu items	Modify menu attributes like path or remove an item	Use hook_menu_alter() in a custom module.
Site settings	Modify default setting	Use hook_strongarm_alter(&$items) in a custom module.
User roles	Modify settings	Use hook_user_default_roles_alter(&$roles) in a custom module.
User roles	Modify settings	Use hook_user_default_roles_alter(&$roles) in a custom module.
CCK fields	Add a new field to an existing content type	Add the new field, and then create a new feature module that includes this field. The exported field will contain information that associates it with the content type.
Views	Modify general settings	Modify with hook_views_pre_build() in a custom module.

immediate, ready or not. Once you make that change, it's live. There is no chance to coordinate other related changes. In addition, all users with access can make a change, even if they didn't mean to make the change. It doesn't require programming, which means it's not in code. This could lead to a couple major issues. First, it's difficult to deploy to a production server. Someone has to write down every configuration change that was made, and then on production, walk back through each change. Second, there's no way to know what state the site is in, so did those changes get made? There's only one way to check—load each configuration page and look.

It's also important to note that if

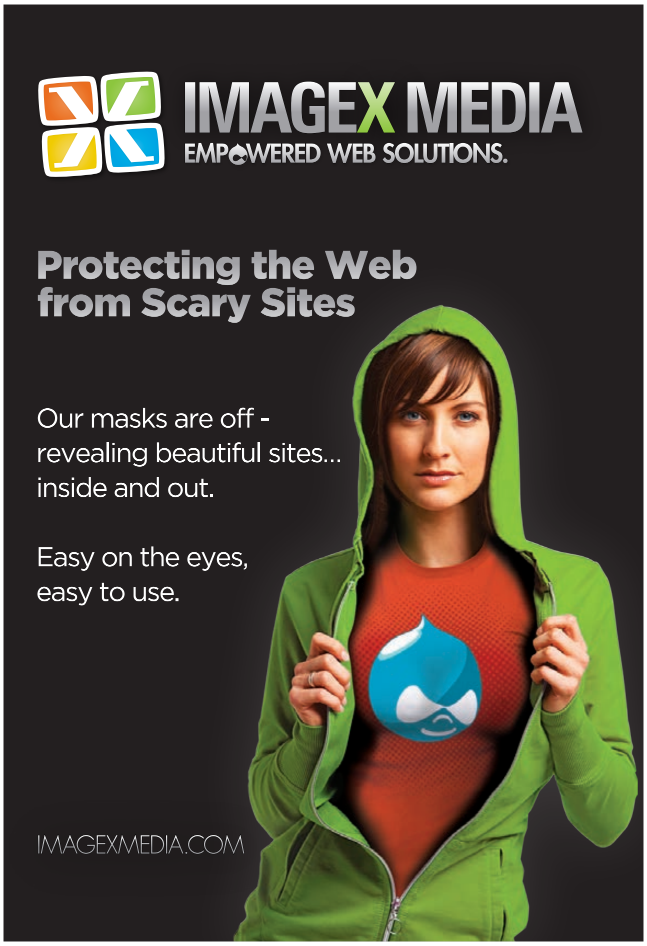
you change a setting through the configuration interface that belongs to a feature module, you may have trouble with future upgrades. When upgrading, you sometimes are required to "revert" the Feature module. This allows the Feature module to make any new changes to its configuration to support the upgrade. When this happens, any setting changes that you made through the configuration interface will be lost.

Custom Override Module

By far, the best way to manage setting overrides in Open Atrium is by making use of a custom override module. This is a simple module that houses a collection of Drupal module hooks and alters that capture the settings into code. This protects your changes from being lost during upgrades and provides these additional benefits:

- Version control: make rollbacks more granular instead of rolling back the entire database.
- Features can be turned on and off per group. Un-featured configurations are global in scope.
- Easily see what has changed from one version of a feature to another using diff tools.
- Use features on other sites, saving countless hours of filling out forms and clicking.
- Features can be shared on feature servers, so you can share your work with the world (or a smaller team).

Creating a custom override module does require a basic understanding of the building blocks of Drupal and Open Atrium, and it requires that you are comfortable with code.


IMAGEX MEDIA
EMPOWERED WEB SOLUTIONS.
Protecting the Web from Scary Sites
Our masks are off - revealing beautiful sites... inside and out.
Easy on the eyes, easy to use.
IMAGEXMEDIA.COM

Something's Missing

You may find that special tools are required for your organization, and although Open Atrium contains a powerful feature set, there's a chance that you'll need something different.

In some cases, you may find that the Open Atrium community already has created the tool you're looking for. For example, a popular tool that often is added to an Open Atrium install is Atrium Folders created by Nuvole. Atrium Folders provides a simple file repository feature. To install this feature, or any other feature or module created by the community, simply download, extract, copy to your /sites/all/modules/contrib folder and enable.

If your needs are even more unique, you can create your own feature or module as a solution. Being built on top of Drupal, Open Atrium includes a wide selection of building blocks contributed by the community, and it often provides an excellent starting point for creating a completely custom feature for your site.

The Next Step

So, you've installed Open Atrium, created a theme, customized Atrium features, added additional features or maybe even built your own. Where do you go from here?

The next step is to get involved directly with the Open Atrium project.

Open source not only means you have the freedom to customize and extend, but also the freedom to participate, and right now is a great time to start. Planning is underway to plot the path and direction Open Atrium will take as it's upgraded to 2.x.

Open Atrium 2.0 Roadmap

We've identified few key areas that we feel should be the focus for Open Atrium 2.0:

- User interface.
- Improved feature set.
- Third-party integration.

Open Atrium's current Ginkgo theme is a vibrant breath of fresh air in the arena of Drupal themes. However, there is room for improvement with its user experience. We are looking to bring a new dynamic theme to Open Atrium 2.x that takes advantage of best practices and technology to provide a world-class user experience, one that fully supports today's mobile devices.

Improved Feature Set

Open Atrium 2.0 also will be improving existing features and adding new features that support the project management workflow. We plan to extend features, such as the Calendar/Task/To-Do feature and Notebook, while also improving their functionality. In addition, we plan to add useful features like Files and Timekeeping.

Additionally, Open Atrium's focus

on project management calls for a shift from a generic "groups" concept to specialized projects. This, coupled with a new ability to assign users to organizations and set privileges around these organizations, could provide users with an intuitive way of managing their projects and team members.

Third-Party Integration

We know a lot of great tools exist that people rely on to get their jobs done. We should work toward building Open Atrium 2.0 with that in mind. Some areas we should look into are calendar, timekeeping and ticketing integration. Of course, a lot of solutions exist, so we're really going to need the community to identify and help build connections to those services.

Implementation and Community Participation

The community is an essential part of any open-source project, and Open Atrium is no exception. Many of these ideas for improvement have come directly from requests through e-mail, Twitter and **https://community.openatrium.com**.

Participation doesn't mean only coding; we want to hear your thoughts and ideas on this plan. However, like all open-source projects, we need developers and designers to help start building the next generation of Open Atrium. This release will not be possible without your coding contributions, testing efforts, documentation assistance and feedback. We are very excited about this next phase of Open Atrium and are excited to work with the community to make it happen! ■

Patrick Settle, Drupal developer, digs into his bag of tricks whenever he runs across a "new" challenge facing Open Atrium, which he's been developing since its beginnings. Thanks to 12 years in system administration for companies from SAIC to the WorldWatch Institute and eight years building Drupal, he's uniquely able to balance a versatility in infrastructure planning with a specialized knowledge of application development that benefits end users and administrators alike.

YOU NEED A MARKETING AGENCY THAT KNOWS THE DIFFERENCE BETWEEN THE **GPL** AND A **GLOBAL PRODUCT LAUNCH.**

We get where you're coming from.
There are a lot of great open-source dev shops out there. But you need one that can speak open-source *and* marketing. If you want a Drupal shop that integrates marketing automation, contextual content and KPIs into a digital strategy designed for Fortune 500 companies, give us a call.

elevatedthird.com
(877) 741-5459

elevatedthird
Digital Agency

Achieving CONTINUOUS INTEGRATION with Drupal

Drupal developers want to follow best practices in software development, testing and deployment, known as **Continuous Integration.** However, they often do not have the time, resources or management support to invest in the necessary infrastructure.

BARRY JASPAN

In the early 1990s, my first job out of college was as a software engineer at a startup company. We were building a commercial product using a well-known open-source network security project. In those days, Agile software development practices (not to mention the World Wide Web, or even widespread public awareness of the Internet) still were in the future. My fellow engineers on that project (who had just graduated with me and to this day are the best programmers I know) and I were taught what we now call the Waterfall method. We thought we were invincible.

We had no idea what was coming. After consultation with potential customers, we wrote a Requirements document describing what the product needed to do, a Functional Specification that described how the product would look and behave, a Design document that described the technical architecture and internals of how we would build it, and even a Test Plan that described the automated tests we would build to ensure the product worked. We had a release deadline, declared by management, of "before Christmas". Good thing we were so young! We engaged in our Death March. The local Chinese delivery place got to know us well. I got home around 1am every morning for months. We finally finished and shipped version 1.0 of the product on December 18. It took me a few weeks to remember what normal humans did when they were not at work.

What Did I Learn from This Experience?

What we did wrong: basically, everything about the software engineering methodology we used was completely stupid. We shipped a working product on time, but we started with the benefit of a working open-source project. We made essentially every mistake that Agile development was invented to prevent.

What we did right: we actually implemented our Test Plan. Since the tests were automated, the build process had to be automated. It certainly added a lot of "extra work" to the project, but the payoff was huge. Before we left for the day, we would kick off the build script. When we came in the next morning, if the last line of output said PASSED, we felt confident and ready to ship. We didn't know it at the time, but we were on the path of what eventually would be called Continuous Integration (CI).

Fast-forward 20 years. I'm now at Acquia, which produces commercial products for companies using the open-source project Drupal. Drupal is a LAMP-stack application for building Web sites and services. We realized early on that everyone using Drupal needs to host it somewhere, and that most people building sites with Drupal do not also want to have to become experts in building a reliable, scalable infrastructure for hosting it. More than that, they also want to be able to follow best practices in software development, testing and deployment; they want to use Continuous Integration. However, they often do not have the time, resources or management support to invest in the necessary infrastructure. I've spent the last three years addressing that problem.

What Is Continuous Integration?

Many excellent and persuasive resources on the Web talk about the principles of CI in detail. In this article, I discuss a simplified list of the most meaningful best practices for Drupal Web site development:

1. *Use a source code repository.* This is step zero for good software development. Most people are doing this, using Git, SVN or other systems; if you are not, start now.

2. *Make small, frequent changes.* All developers should commit their changes frequently. This reduces the inevitable conflicts and lets problems surface sooner. Also, small, frequent changes enable small, frequent releases, making all the rest of the principles more valuable.

3. *Automate testing.* Have your repository automatically integrated with a testing environment, so that every commit triggers a test run. This way, you know immediately if something broke.

4. *Test in a clone of the production environment.* It does no good to test your software under different conditions from those that it will run in production; doing so is a recipe for taking down your site when you deploy. Never hear someone say "But it worked on my machine!" again.

5. *Make all versions easily accessible.* Despite best efforts, production releases still will break, so you need an easy way to re-deploy a prior version. Then, you'll want to compare the working and broken versions to figure out what went wrong. To do this, you'll need a reference copy of past releases.

6. *Have an audit trail (that is, a blame list).* This helps you not just in the source control of who made this commit, but who deployed the commit as well. This can provide rationale as well as potential fixes.

7. *Automate site deployment.* In order to tolerate small, frequent releases, pushing a release needs to be an automated process so it's very quick and easy. If it's a big chore to push one release, the whole process falls apart.

8. *Measure results and iterate rapidly.* Are the changes helping? Is the site faster? Did the usability enhancement yield more sales? If it's not, you can iterate again.

Achieving Continuous Integration requires some amount of infrastructure, the culture and discipline of the engineering team to use it, and management's understanding and commitment so that it supports the necessary investment. This is an article about technology, not management and culture, so I focus primarily on the infrastructure here.

Building It Yourself

Many shops build their own CI systems that are perfectly tailored to their own needs. Doing so is perfectly reasonable if you have the time and resources to get there. The biggest danger of doing it yourself, of course, is deciding to—and then not getting around to it. You end up doing things the manual, slow and error-prone way "until we have time to fix it", which often turns out to be "never". When you do get started, it probably will end up being a permanent side project, which may lead you to cut corners that will end up causing problems at the worst possible time later.

Here are some of the things you should keep in mind.

Use a source code repository. You probably already are (right?). You will need to be familiar with its "post-commit hook" capability to script actions based on it. If you are using a hosted repository (such as GitHub), you will have to integrate with its Web-based hooks.

Make small, frequent changes. All of your developers will be making frequent commits, resolving conflicts locally as best they can. To keep things moving forward, you need to have a constantly available running copy of everyone's latest code. One way to do this is to deploy the tip of your main development branch automatically to a shared development environment, so everyone always can see it. You can script this yourself using your repo's post-commit hooks. A build automation tool like Jenkins will help, but you still need to write the deployment script yourself.

Automate testing. Assuming you write automated tests for your site, you will want to run them every time someone makes what they believe is a release-ready commit. Lots of tools exist for doing this. One popular choice is Jenkins (formerly called Hudson), and it is excellent. It can integrate directly with your code repository and trigger a "job" on every commit, or run a job on a schedule.

IF YOU FOOL YOURSELF THAT YOU CAN "MOCK OUT" THESE DEPENDENCIES AND HAVE PURELY STANDALONE UNIT TESTS THAT CAN RUN ANYWHERE, REALITY WILL MOCK YOU BACK.

The tests themselves are not the whole story though. Because your application is a Drupal site, you need to test it in a Web environment. You'll certainly need a running database server. If you want to test actual page loads like a browser would see, you'll need a running Web server too. You probably want to test your application along with a reasonably current production database; if you don't automate that, one day you'll find yourself testing against year-old data. However, you also probably want to "scrub" your current production database before running tests against it, lest you accidentally spam all your customers from your test servers, or worse. This is all the responsibility of your test harness script, run by Jenkins.

If you fool yourself that you can "mock out" these dependencies and have purely standalone unit tests that can run anywhere, reality will mock you back. You will discover that tests are not accurately simulating your live environment, and you will have to roll back a release that "passed all of its tests" but failed in production.

Test in a clone of the production environment. This is where things really get interesting. I've already talked about needing a running Web and database server. If your site uses additional services like memcached, Varnish or Apache Solr, you need to make sure those are in place too. If your production site uses SSL, you either need SSL running in your testing environment, or you need to turn off the checks or redirection that enforces it. Ultimately, it is as much work to maintain your test environment as it is your production environment.

Where do you run all this stuff? The "simple" answer is to run it on the same server as Jenkins. However, Jenkins probably is not running on your production servers, so immediately your test environment is different from production. Do you *know* that when you install Jenkins via your distro's package manager that it does not pull in some other package that your site might end up using in testing but then fail because it is missing in production?

This points to an even deeper issue. You cannot create a clone of your production environment unless you know exactly what your production environment really is. What packages are installed? What configuration files are in place? What dæmons are running? What security updates have been installed? Running a production Web site leads to all kinds of unexpected issues and surprises, and even the best-intentioned, well-meaning sysadmins are likely to solve a crisis by changing some configuration on the server by hand. You have to make sure those changes always get propagated to your test environment. For that matter, you have to make sure they are permanently maintained in your production environment too.

This leads directly to the topic of DevOps and server configuration management. The only way to be sure your production environment is the way you expect is to automate its configuration, and the only way to ensure that your test environment is a clone of your production environment is to use exactly the same automated configuration to build it. There are good open-source tools for doing this; Puppet and Chef are the two I am familiar with. However, Puppet and Chef are programming languages in their own right. Once you go down this path, you are now maintaining two completely different pieces of software: your Web application and the infrastructure automation to run it. At this point, you need to make a recursive call to start reading this article over at the beginning, because you will need to use Continuous Integration on your infrastructure automation just like you do for your Web app. So, your Web app needs a production and test environment, all of which is running in your production infrastructure environment; now you need a test infrastructure environment in which to test updates to your infrastructure code before rolling them out to production. If you are using Jenkins to run your CI process, and Jenkins is deployed as part of the infrastructure you are developing, then...your brain just hit a stack overflow and exploded. Ooops.

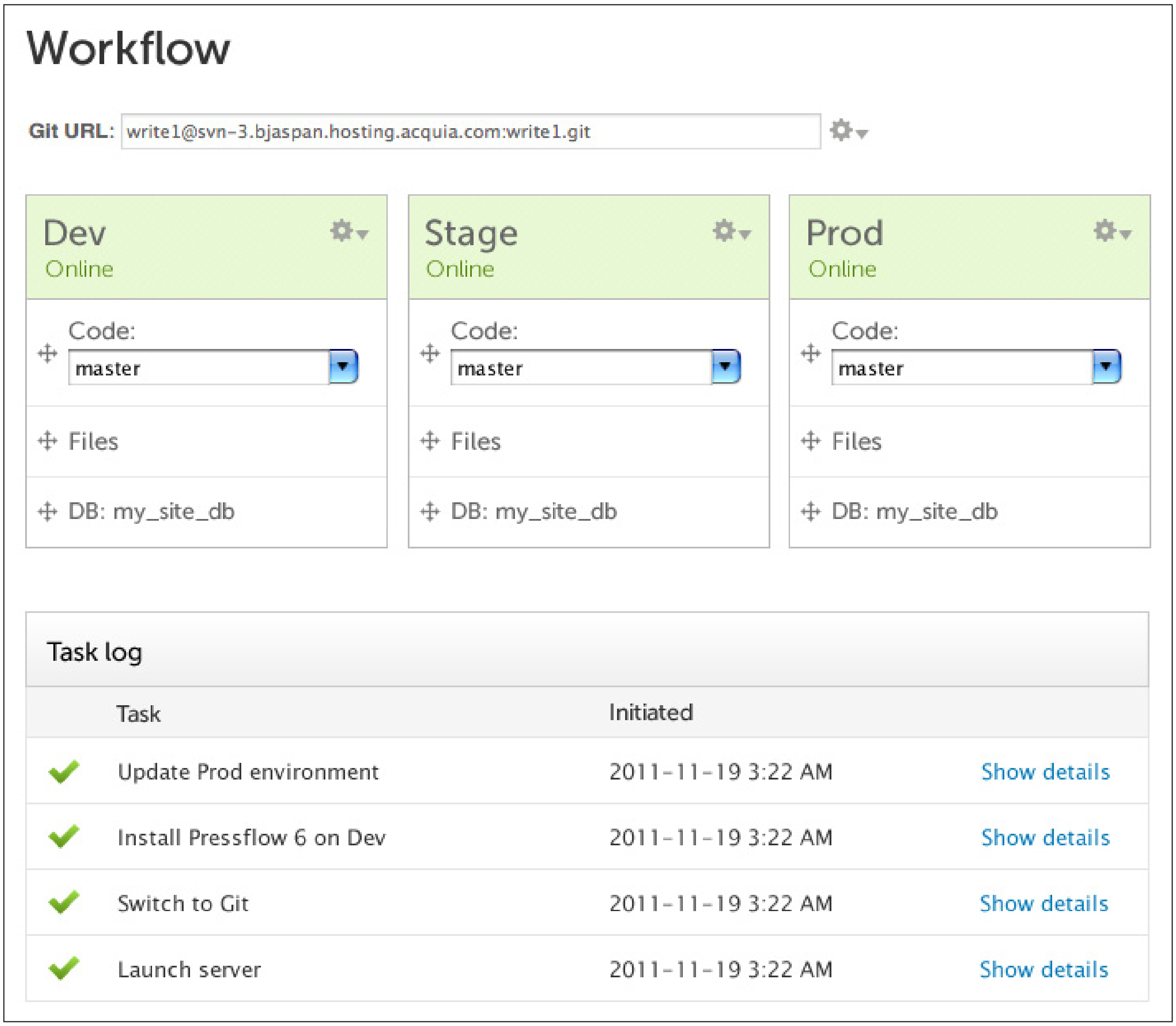
To be clear, this is all doable, and there may be simplifying assumptions you can make to reduce the effort. However, if you make the mistake of thinking of your server configuration as something you can just build once and forget about, your Web site is eventually going to suffer for it.

Make all versions easily accessible. When it's time to push to production, you want to create a symbolic tag in your version control system that says what you released when. If you release frequently, you'll end up with a lot of tags, but that's okay; they're cheap. You probably will end up creating these tags in the script you create to automate deployment.

Maintain an audit trail. Your VCS gives you a commit history for your source code, but you need more than that. When something goes wrong, you easily should be able to point to the date/time/individual that played a part and quickly get the information you need. Who pushed the release to production earlier today? Who added a new domain name to Apache virtual host configuration? Can you verify that the SSH key for the employee that left last week has been removed? Most changes will be in your Web site source code, but some will be in your infrastructure configuration code, so you will want a unified view of the changes.

Automate site deployment. Okay, so you are working in small batches with frequent commits, testing every time in a clone of the production environment. Now it needs to be easy to push your new application to the live environment. If you've automated your infrastructure as described and already have a system in place for deploying new code commits to your testing environment, this should be a pretty small additional step. It has to be simple, fast and reliable; you want to be able to push a release and go have lunch five minutes later without worrying about it.

Measure results and iterate rapidly. There are many great monitoring and measurement tools available that check for things, such as error logs, page load performance, server performance, A/B testing and more. Because



	Task	Initiated	
✓	Update Prod environment	2011-11-19 3:22 AM	Show details
✓	Install Pressflow 6 on Dev	2011-11-19 3:22 AM	Show details
✓	Switch to Git	2011-11-19 3:22 AM	Show details
✓	Launch server	2011-11-19 3:22 AM	Show details

Figure 1. The Workflow page is the centerpiece of Acquia Cloud's CI system.

you've automated your infrastructure configuration, integrating these onto your servers is not that much additional work, but you still have to decide which ones to use, how best to install them, and how to get the data you need out of them most efficiently.

Use an Existing System

Whew! Okay, be honest. How likely is your company actually to make the investment to build and deploy an automated CI infrastructure as described above? I thought so. The fact is that infrastructure is not your specialty. (If it is, can we hire you?) You build exceptional Web sites, and you should not spend so much time and effort also building the servers to run it.

Your alternative is to use a system someone else built for you. Several exist, each with different properties, and with more arriving all the time. I happen to be the lead engineer for Acquia Cloud, so

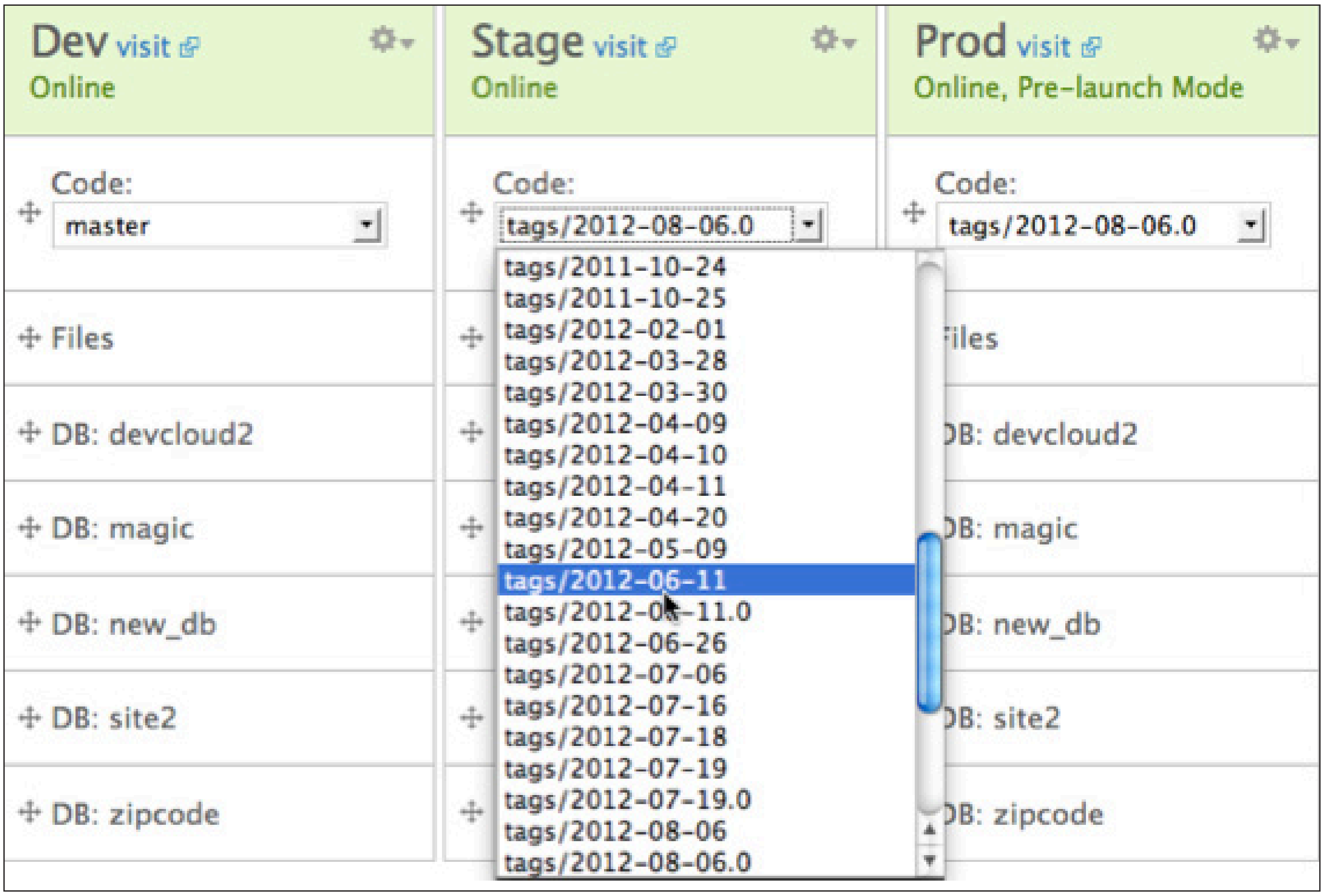


Figure 2. The Code selector lets you deploy any branch or prior release version to any environment.

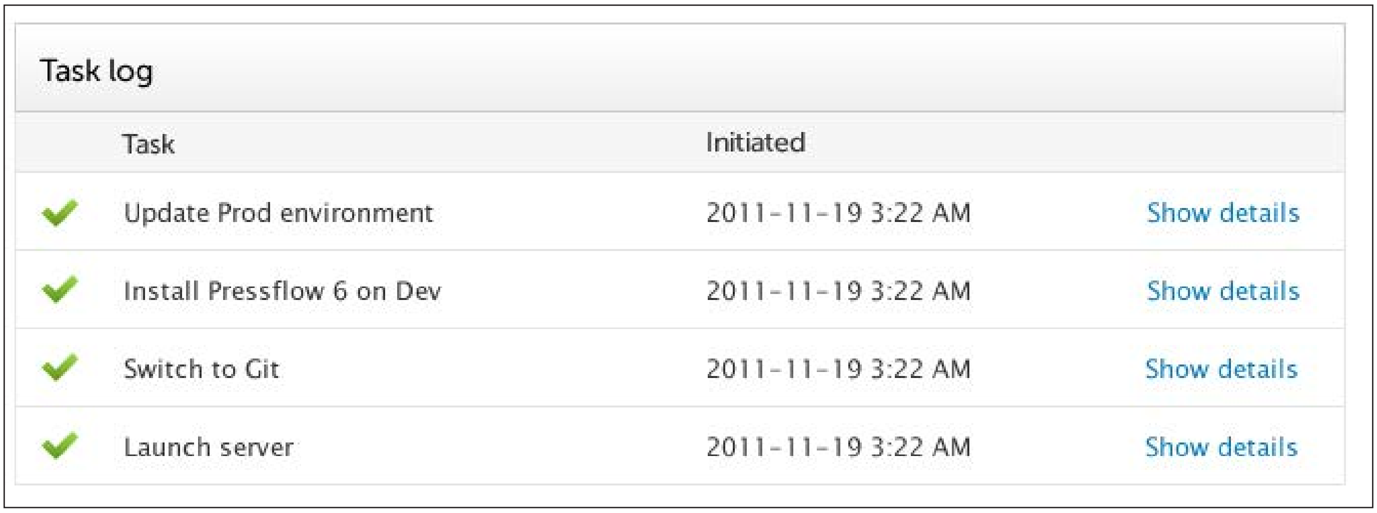
ACQUIA CLOUD PROVIDES A DEVELOPMENT, STAGING AND PRODUCTION ENVIRONMENT FOR YOUR SITE.

let me quickly demonstrate how Acquia Cloud provides everything you need to implement CI for your Web site.

Use a source code repository. Acquia Cloud provides both Git and SVN repositories. The URL is displayed right at the top.

Make small, frequent changes. Acquia Cloud provides a development, staging and production environment for your site. You can deploy any branch or tag from your repository in any of them. When you deploy a branch in an environment, every commit to that branch is deployed to that environment. This makes the Dev environment perfect for initial integration testing. Set it to deploy the "master" (Git) or "trunk" (SVN) branch, and every developer's commits are available immediately for initial experimentation.

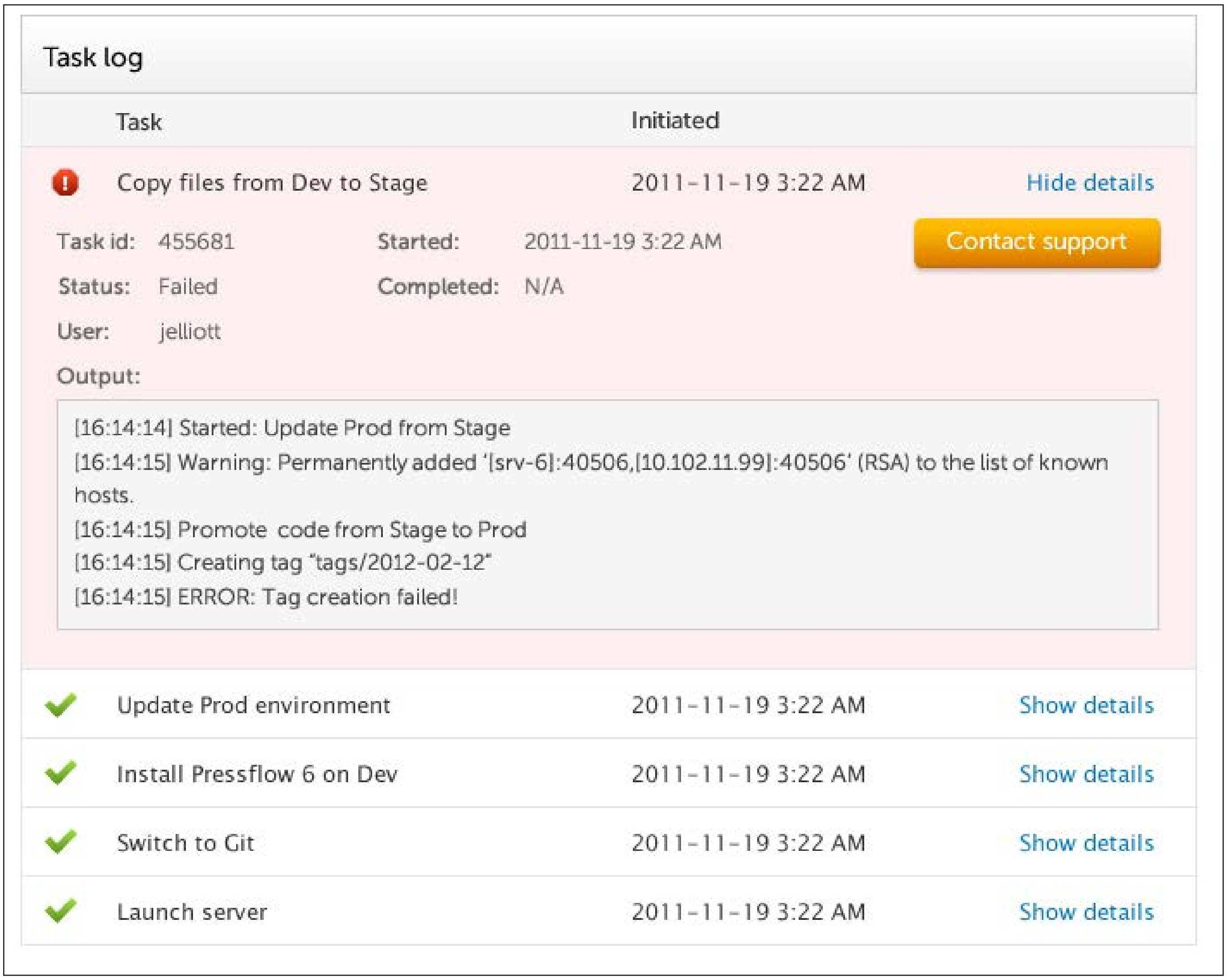
Automate testing. Every time you deploy code or perform various other actions, Acquia Cloud runs "Cloud Hooks". These are simple scripts that you put into your code repository to perform any actions you want. Each hook is tied to specific actions in a specific environment—for example, all scripts in the hooks/post-code-deploy/prod directory of your repository run when you deploy code to the



Task log

	Task	Initiated	
✓	Update Prod environment	2011-11-19 3:22 AM	Show details
✓	Install Pressflow 6 on Dev	2011-11-19 3:22 AM	Show details
✓	Switch to Git	2011-11-19 3:22 AM	Show details
✓	Launch server	2011-11-19 3:22 AM	Show details

Figure 3. The Task log shows all changes to any of your site's environments.



	Task	Initiated	
❗	Copy files from Dev to Stage	2011-11-19 3:22 AM	Hide details

Task id: 455681 Started: 2011-11-19 3:22 AM
Status: Failed Completed: N/A
User: jelliott
Output:

```
[16:14:14] Started: Update Prod from Stage
[16:14:15] Warning: Permanently added '[srv-6]:40506,[10.102.11.99]:40506' (RSA) to the list of known
hosts.
[16:14:15] Promote  code from Stage to Prod
[16:14:15] Creating tag "tags/2012-02-12"
[16:14:15] ERROR: Tag creation failed!
```

	Task	Initiated	
✓	Update Prod environment	2011-11-19 3:22 AM	Show details
✓	Install Pressflow 6 on Dev	2011-11-19 3:22 AM	Show details
✓	Switch to Git	2011-11-19 3:22 AM	Show details
✓	Launch server	2011-11-19 3:22 AM	Show details

Figure 4. Details are available for every task.

Production environment. The hook scripts run in sorted order until the first one fails, and the output from all hook scripts is available at the end. This is the perfect way to run your test scripts, scrub a database, perform a load test or anything else.

Test in a clone of the production environment. This is the biggest payoff with Acquia Cloud. We maintain each of these environments—Dev, Stage and Prod—for you. You can choose whether they are on the same or different servers, and whether they are redundant and load balanced or running on a single VM, but we ensure that as far as your Web application is concerned, the configuration is identical. Of course, we also provide 24/7 monitoring, backups, security updates and critical fixes—all

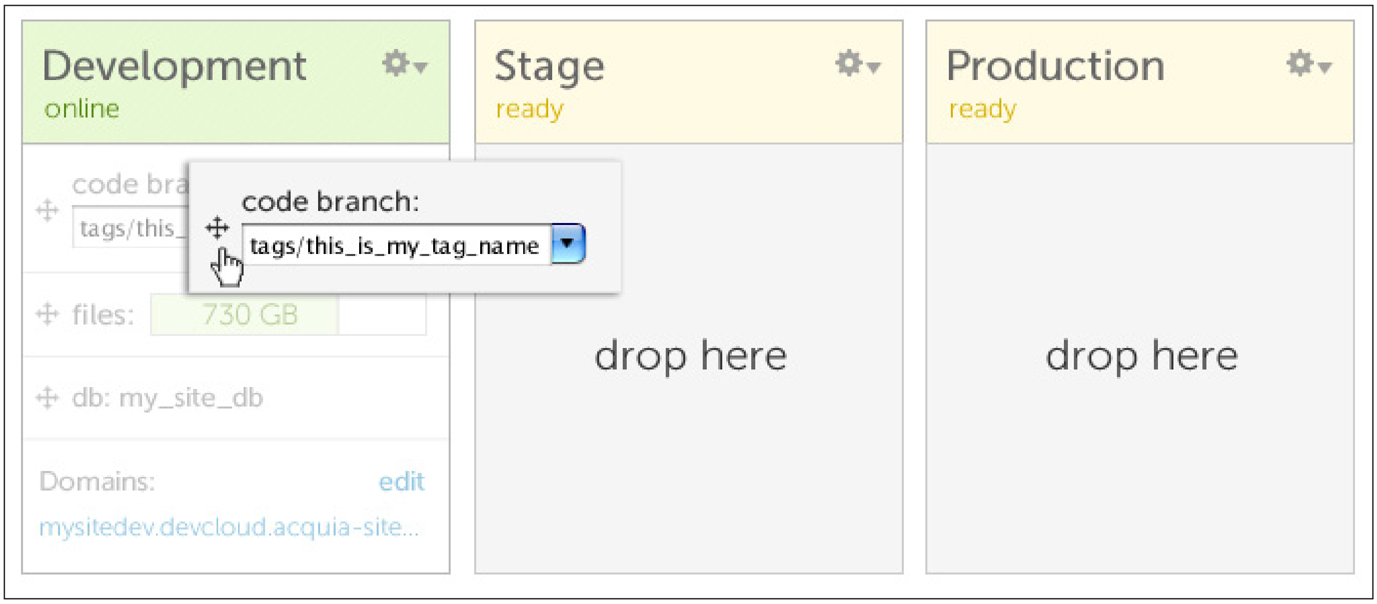


Figure 5. Deploying code is a simple drag-and-drop operation (API and CLI also available).

the things you would have to do on your servers yourself.

Make all versions easily accessible. As you can see in Figure 2, you can always revert back to any specific tagged version or branch in any environment.

Have an audit trail (that is, a blame list). Our task log is your audit trail. It shows code commits, but also all changes to your Web environment: domain names, SSH keys, server launches and so on. It shows you exactly what date and time each action took place, with the option to show the full detail for the command.

Automate site deployment. To deploy a release from one environment to another, simply drag and drop on the UI (or use our API or Drush CLI to do the same thing). If you drag code from an environment deploying a branch, it creates a symbolic tag at the tip of that branch and deploys the tag in the target environment. If you drag an environment deploying a tag, it just deploys the same tag in the same environment. You always can deploy any branch or any previous tag in any environment just by selecting it from the drop-down list (or, again, via our API or CLI).

Measure results and iterate rapidly. This article mostly been has about Acquia Cloud, but Acquia Cloud is itself just a feature of the Acquia Network that provides a wide variety of tools to improve your site, such as expert Drupal configuration advice, SEO optimization, faceted search, performance monitoring, load testing and spam blocking, plus

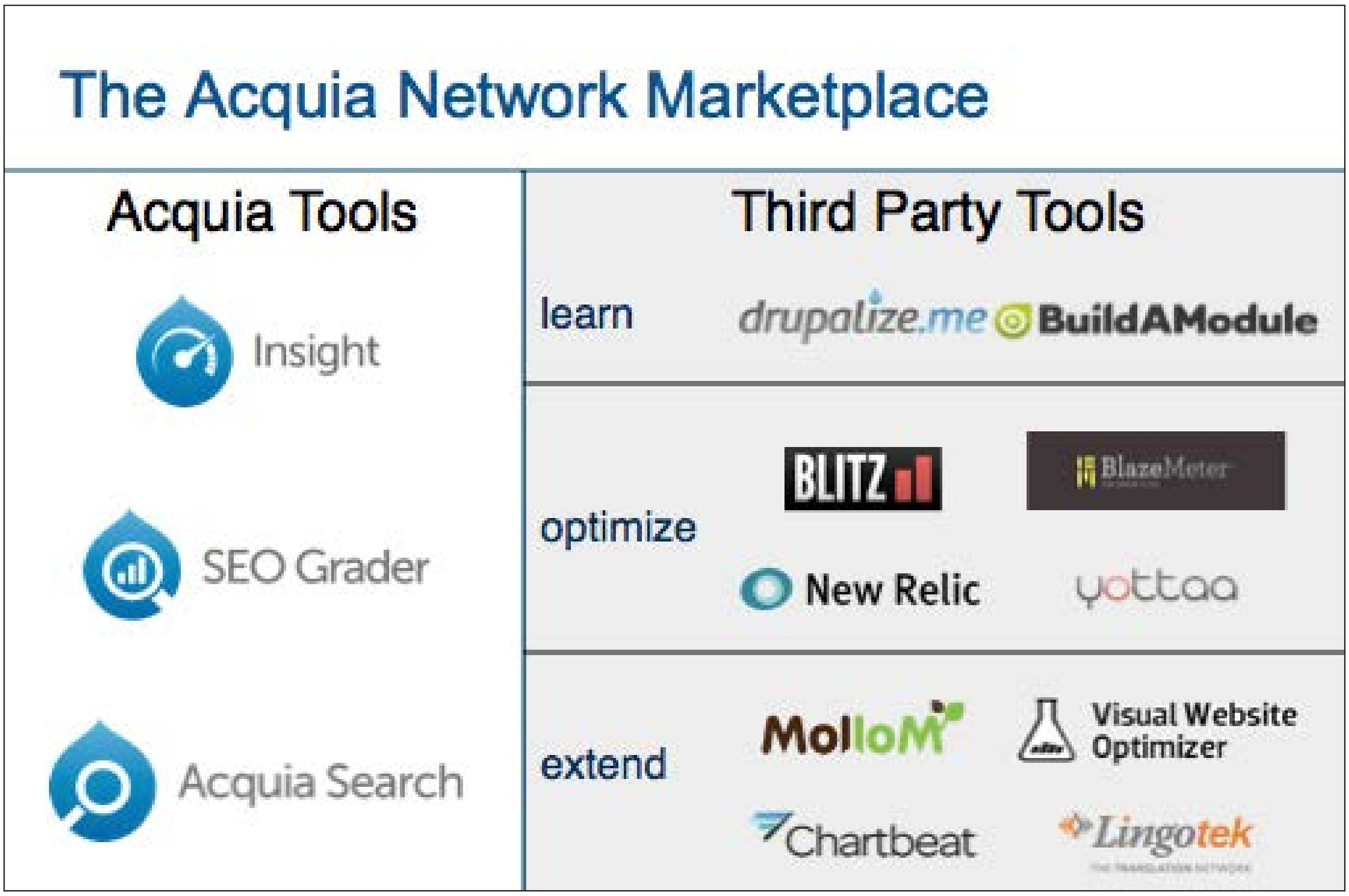


Figure 6. The Acquia Network provides resources to understand and improve your site's results.

services like education, training and support. The Acquia Network is part of every Cloud subscription and includes all of these tools, most for free.

To view a video overview that showcases how to develop with Acquia Cloud for your Drupal site, I have a Webinar on this very topic available at **http://ow.ly/cUNlL**. To sign up for a completely free version of Acquia Cloud, visit **http://network.acquia.com/freecloud**. ■

Barry Jaspan is a serial software engineer and entrepreneur who has been creating and selling open-source software products literally since he was 12 years old (many moons ago!). Currently Senior Architect at Acquia, he leads Acquia Cloud, the Drupal-optimized PHP cloud hosting platform that runs tens of thousands of Drupal Web sites on thousands of cloud-based servers for some of the best-known brands in the world. He is an angel investor and advisor to Boston-area startups and, when he pulls himself away from the keyboard, an avid whitewater kayaker and rock climber.

RedHen CRM an Open-Source CRM Solution Built Entirely with Drupal

Historically difficult to implement in Drupal, native CRM solutions in Drupal 7 now allow site builders to craft more personalized user experiences for their Web site visitors by integrating their Drupal CMS with their customer relationship management system (CRM). SEAN LARKIN and LEV TSYPIN

One of Drupal's key value propositions is its ability to integrate conventional content management features with other Web-based applications and tools. More and more, Drupal is becoming a Web application platform for building engaging Web sites that provide rich, personalized content experiences for site visitors.

We often recommend that our clients begin crafting more personalized user experiences for their Web site visitors by integrating their Drupal CMS with their customer relationship management system (CRM).

The benefits of such integrations are many. You can present forms for your site visitors to update their own CRM contact records or to provide you with better data about their interests and communication preferences. Based upon such information, you can serve up more pertinent content, as well as offer premium access to special features.

In fact, the value of CRM/CMS integration is becoming so compelling that most large CRM companies, such as Salesforce, Microsoft Dynamics, Oracle and Blackbaud, are enhancing their Web-based products and adjusting their marketing language to describe their offerings as "social enterprise" or "customer engagement" platforms. The pundits often use the buzzwords "Social CRM" to describe these retooled and rebranded CRM solutions.

The State of Third-Party CRM Integration in Drupal

Drupal has long been ahead of the CRM integration curve because of its open-source flexibility. Integration between Drupal and the open-source CRM CiviCRM (**http://civicrm.org**) historically has seen the widest adoption. Like Drupal, CiviCRM runs on a LAMP stack (Linux, Apache, MySQL and PHP) and can be configured to run as a component of a Drupal Web site. CiviCRM is a mature solution with a number of great features mostly geared toward the constituent relationship management needs of nonprofit organizations.

While CiviCRM's integration with Drupal is robust, it's still not perfectly seamless due to the fact that CiviCRM was engineered to work with multiple CMS front ends. CiviCRM relies on its own database schema and APIs. It also leverages its own templating engine that presents different markup from that generated by Drupal. For Drupal developers looking to build tightly integrated CMS/CRM solutions, CiviCRM integration requires mastering more APIs and maintaining more code. For site builders and Web site administrators, CiviCRM integration requires additional training, because CiviCRM's administrative interfaces follow different design patterns from those provided by Drupal. Further, given that major releases of CiviCRM rely upon specific versions of Drupal core, upgrading Drupal-CiviCRM solutions can present significant hurdles.

CRM Options Built Natively within Drupal

The Drupal community has long aspired for native CRM functionality. The most obvious benefits of native CRM in Drupal include:

- A more seamless user experience for site visitors registering for events, making donations/payments or engaging in other Web site transactions.
- The opportunity to leverage Drupal's growing suite of mobile and responsive tools and themes for CRM interfaces.
- The ability to expose CRM data as content and display aggregate CRM data on your site.
- Increased opportunities to integrate CRM data with Drupal contributed modules, such as data visualization and geo-mapping.
- Reduced staff training costs by eliminating the need for staff training on multiple platforms.
- Potential reductions in technical risk since all your tools rely on a single Drupal instance.
- Potential reductions in hosting and IT costs.
- The ability to do complex engagement scoring or engagement analytics (more on this later).
- And probably most important, the ability to customize fully your CRM solution *"the Drupal way"*.

The arrival of the *entity framework* in Drupal 7 core has opened the door to the development of more robust CRM solutions built natively in Drupal. So, after a year of scheming and more than 1,000 hours of intense development, ThinkShout, Inc., recently announced the beta release of RedHen CRM—a native CRM solution written *for Drupal in Drupal*. And, we anticipate that RedHen will have a stable release by the time this article is published.

RedHen CRM as a CRM "Framework"

ThinkShout initially designed RedHen CRM around the complex association management (AMS) needs of nonprofits and trade associations. Based upon a site visitor's affiliation with an organization that has purchased one or more membership subscriptions, that site visitor is provided premium access to content, e-commerce discounts and advanced Web site features when logged in to a Web site built on top of RedHen CRM.

These association management requirements go well beyond the standard needs of most CRM/CMS solutions. However, in building an abstract tool with these needs at the forefront, our goal has been to future-proof RedHen CRM as much as possible to ensure that its architecture can accommodate a wide variety of use cases.

RedHen CRM is similar to Drupal Commerce (**http://drupal.org/project/commerce**) in its modular structure. As with Drupal Commerce, the core RedHen modules that can be downloaded on the drupal.org project page won't provide you with a working CRM right out of the box. RedHen is intended to provide developers and site builders with the building blocks for quickly creating their own CRM data models that map to their particular business requirements and workflow. Consequently, setting up a new RedHen CRM instance does require configuration.

In the future, ThinkShout is likely to release RedHen "Features" or "Apps" that provide prepackaged CRM solutions for different use cases. At the time of this writing, we know of more than 96 RedHen CRM solutions built by other Drupal developers and site builders, and we anticipate this number will increase by the time this article is published.

In addition to its association management uses, we see RedHen CRM as an ideal starting point for building custom sales pipeline management tools and project management applications, as well as Drupal integration points with third-party ERP (enterprise resource planning) tools and financial accounting packages.

RedHen CRM Project Structure

RedHen CRM consists of a core module containing shared APIs and interfaces and a collection of feature-specific sub-modules, including:

- Contact (redhen_contact): contact entities and APIs, along with integration with Drupal users.
- Fields (redhen_fields): custom field types used by RedHen entities. Currently this includes a unique e-mail field, which assigns attributes to e-mails, such primary, bulk and custom labels (home, work and so on).
- Organization (redhen_org): organization entities and APIs.

- Organization Group (redhen_group): lightweight group feature that turns organizations into containers for private content with a broadcast system.

- Relations (redhen_relation): the Relation module allows connections between contacts and organizations.

- Note (redhen_note): notes on contacts and organizations.

- Engagement (redhen_engagement): engagement scoring system and APIs. RedHen Note integration is included, as well as Rules integration for popular modules, such as Comment, Registration and Webform.

- Registration (redhen_registration): integration with the Entity Registration module.

We maintain other functionality that is not needed for all uses cases in separate projects in order to keep the core RedHen code base as lean as possible. The RedHen Membership system (**http://drupal.org/project/redhen_membership**) is our most widely used RedHen component with its own project name space. It handles individual and organizational membership subscriptions. As with Drupal Commerce, key sub-modules will continue to be included with the main module code base. However, we anticipate that as the RedHen CRM developer community grows, we will see more and more contributed modules that extend RedHen's core feature set.

The overall architecture of RedHen CRM consists of a set of minimalistic building blocks that developers and site builders can use to develop solutions tailored to specific use cases. Like Drupal Commerce, RedHen relies on Drupal distributions and installation profiles for the heavy lifting of fleshing out polished applications that serve specific needs.

RedHen CRM Module Dependencies

RedHen has minimal dependencies on other Drupal contributed modules, striving for a balance between relying on its own code base and leveraging other contributed modules. Beyond Drupal core, RedHen CRM currently depends only on Entity API, discussed in detail below and the Relation module (**http://drupal.org/project/relation**) for managing connections. RedHen CRM *supports* integration with popular contributed tools, like Views and Rules, but these modules are not required. This makes RedHen a leaner and more stable application platform, as it relies less on a shifting foundation of other contributed modules over which we have little to no control.

This centrist approach helps assure other Drupal developers that their customizations and extensions of RedHen won't break due to fragile module interdependencies. At the same time, it provides less-technical site builders with a known approach to extending RedHen CRM with standard Drupal tools, such as Views, Rules and the Fields API.

Custom Entities

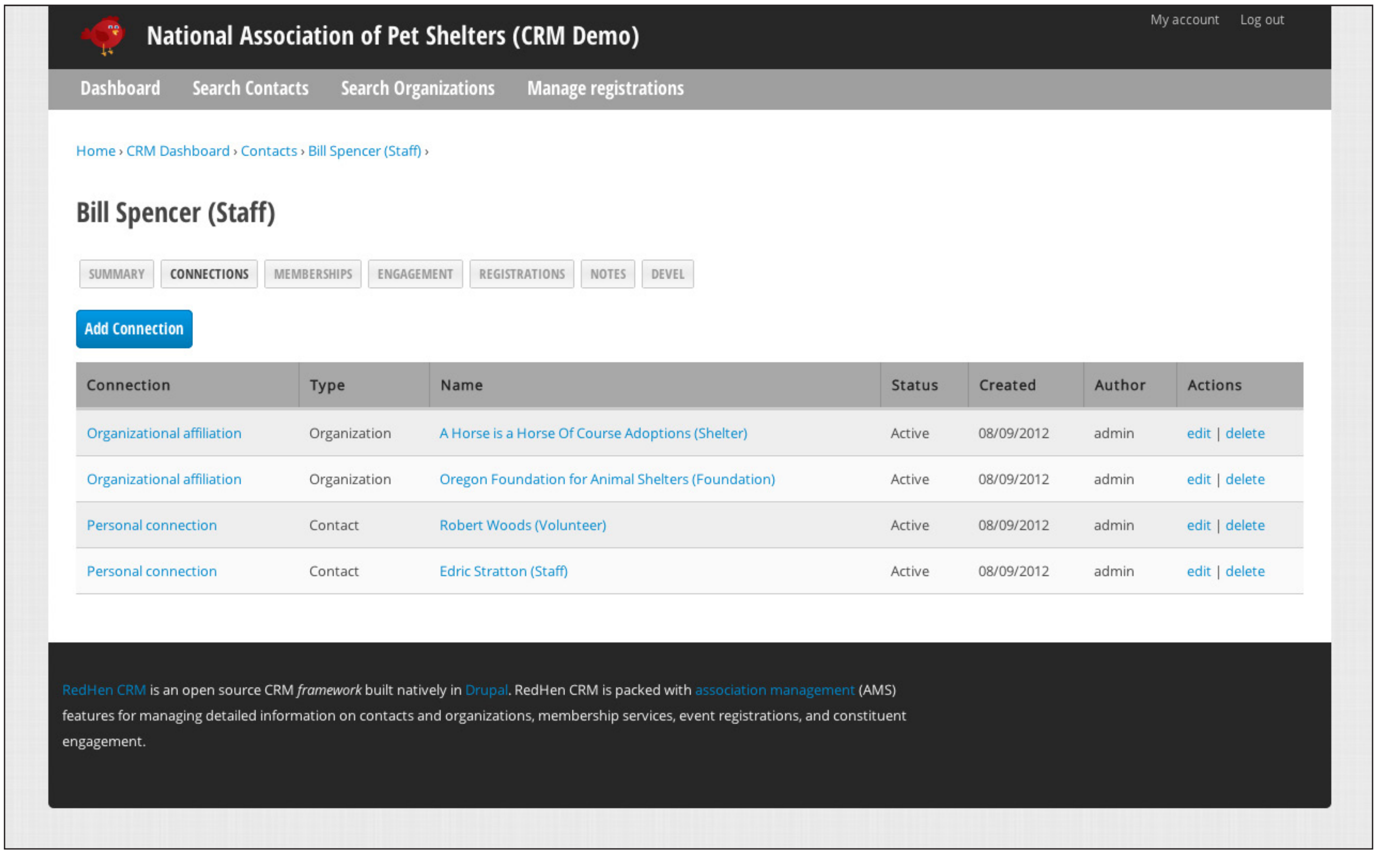
RedHen CRM relies on custom Drupal entity types and bundles. It leans heavily on Entity API (**http://drupal.org/project/entity**), a wrapper around Drupal's core entity system that eases developing entities in several ways. The Entity API module:

- Provides classes and controllers to streamline entity creation and management. RedHen extends these base classes as needed.
- Eases integration with key contributed modules, namely Views (**http://drupal.org/project/views**) and Rules (**http://drupal.org/project/rules**).
- Exposes standardized API hooks during CRUD operations providing a consistent interface for other modules to interface with your custom entities.
- Provides hooks and data structures allowing for entities to be exported, for example, using Features or CTools. RedHen leverages this feature to make entity bundle definitions, such as types of contacts and organizations, exportable.

NOTE: At the time of this writing, Entity API has 138,685 reported installs all on Drupal 7 and is among the top 20 modules on drupal.org. As Drupal dependencies go, it is highly stable and reliable.

RedHen ships with the following entity types, each of which comes bundled with core properties and can be extended with additional user-defined fields:

- Contacts
- Organizations
- Notes
- Memberships (part of RedHen Membership)
- Engagements



Bill Spencer (Staff)

Connection	Type	Name	Status	Created	Author	Actions
Organizational affiliation	Organization	A Horse is a Horse Of Course Adoptions (Shelter)	Active	08/09/2012	admin	edit | delete
Organizational affiliation	Organization	Oregon Foundation for Animal Shelters (Foundation)	Active	08/09/2012	admin	edit | delete
Personal connection	Contact	Robert Woods (Volunteer)	Active	08/09/2012	admin	edit | delete
Personal connection	Contact	Edric Stratton (Staff)	Active	08/09/2012	admin	edit | delete

RedHen CRM is an open source CRM *framework* built natively in Drupal. RedHen CRM is packed with association management (AMS) features for managing detailed information on contacts and organizations, membership services, event registrations, and constituent engagement.

Figure 1. Listing of a Contact's Connections

Connections

A key feature of any CRM is managing connections between organizations and contacts. RedHen features a very flexible connection system built on top of the Relation module, which allows for bidirectional connections between arbitrary types of entities. These relationships themselves can have additional user-defined *fields*. RedHen ships with two types of relationships: Affiliations, which are connections between an organization and a contact, and Personal Connections, which are connections between contacts. These relationships come bundled with status and role properties, but fields can be added that are applicable to a given relationship.

For example, imagine that you want to define a relationship between a contact of the type "Staff" and an organization of the type "Company". Suppose you want to include information about the position that contact has at that organization. This field value has meaning only within the context of relationship, and therefore, this data is stored with the relationship rather than on the contact record or the organization record itself.

Building an Example Application with RedHen

Now that you've learned about RedHen, you most likely will want to know the steps involved in building a useful application with it. Below is a step-by-step tutorial for setting up an association management system (AMS) for the National Association of Pet Shelters, an fictional nonprofit organization. For those who want to follow along, the complete demo application is available for download as an installation profile called RedHen Demo (**http://drupal.org/project/redhen_demo**) on drupal.org.

Installation

Here are the installation steps:

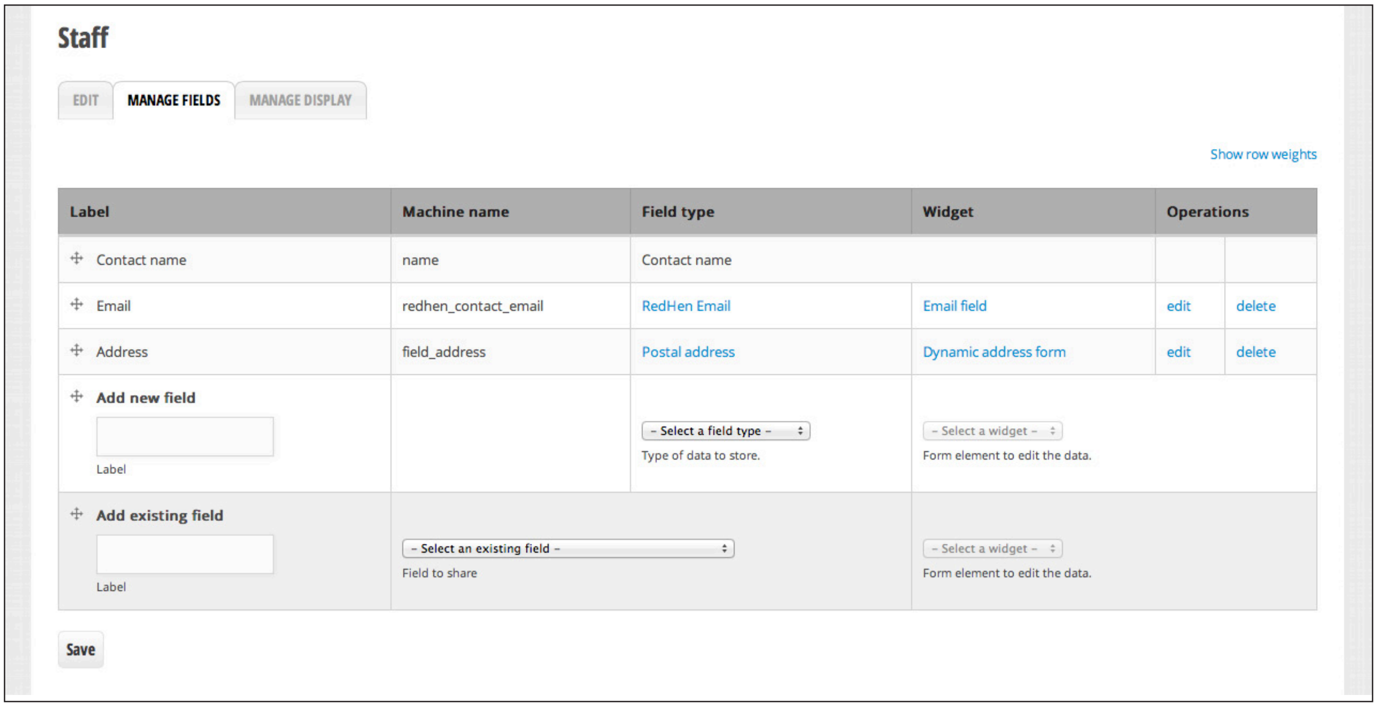
1. Download the latest version of Drupal 7, RedHen, RedHen Membership, Entity API, Relation and Views.

2. Install Drupal using an installation profile of your choice.

3. Enable RedHen and its dependencies.

Structure

When RedHen is enabled by itself, it defines *types* of entities, but it does not



Staff

EDIT | **MANAGE FIELDS** | MANAGE DISPLAY

Show row weights

Label	Machine name	Field type	Widget	Operations	
Contact name	name	Contact name			
Email	redhen_contact_email	RedHen Email	Email field	edit	delete
Address	field_address	Postal address	Dynamic address form	edit	delete
Add new field Label		- Select a field type - Type of data to store.	- Select a widget - Form element to edit the data.		
Add existing field Label	- Select an existing field - Field to share		- Select a widget - Form element to edit the data.		

Save

Figure 2. Managing the Fields Associated with a Contact Record

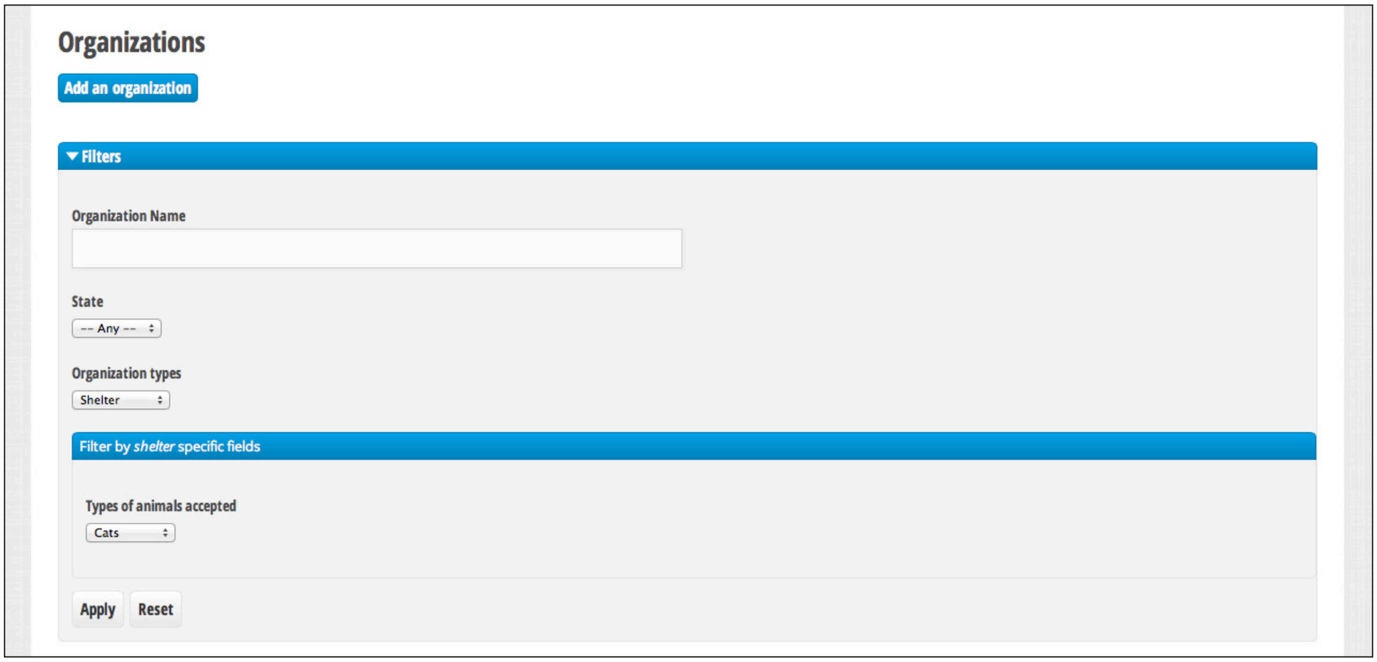
create *instances* of these entity types. That is because, out of the box, RedHen doesn't presume to know the types of contacts, organization, memberships and so on, that will be needed for a given CRM deployment. In Drupal speak, these entity type instances are called bundles. In the next step, you will define bundles for each of your RedHen entity types.

First, navigate to /admin/structure/redhen to see the types of entities that can be managed through RedHen. They are, again, contacts, engagement scores, memberships, notes and organizations. For each of them, create the following bundles, respectively:

- Contacts: contacts have only a label and machine name. Create one for Staff and Volunteers.

- Organizations: aside from machine name and label, organization bundles can be turned into content groups if the redhen_org_group module is enabled. Checking the "groupify" box exposes two additional settings: private and a list of content types that can be posted into the group. Create two organizations, Foundation and Shelter, making the latter a content group.

- Notes: notes come with a default single bundle since there is only one type of note.


Organizations

Add an organization

Filters

Organization Name

State: -- Any --

Organization types: Shelter

Filter by *shelter* specific fields

Types of animals accepted: Cats

Apply Reset

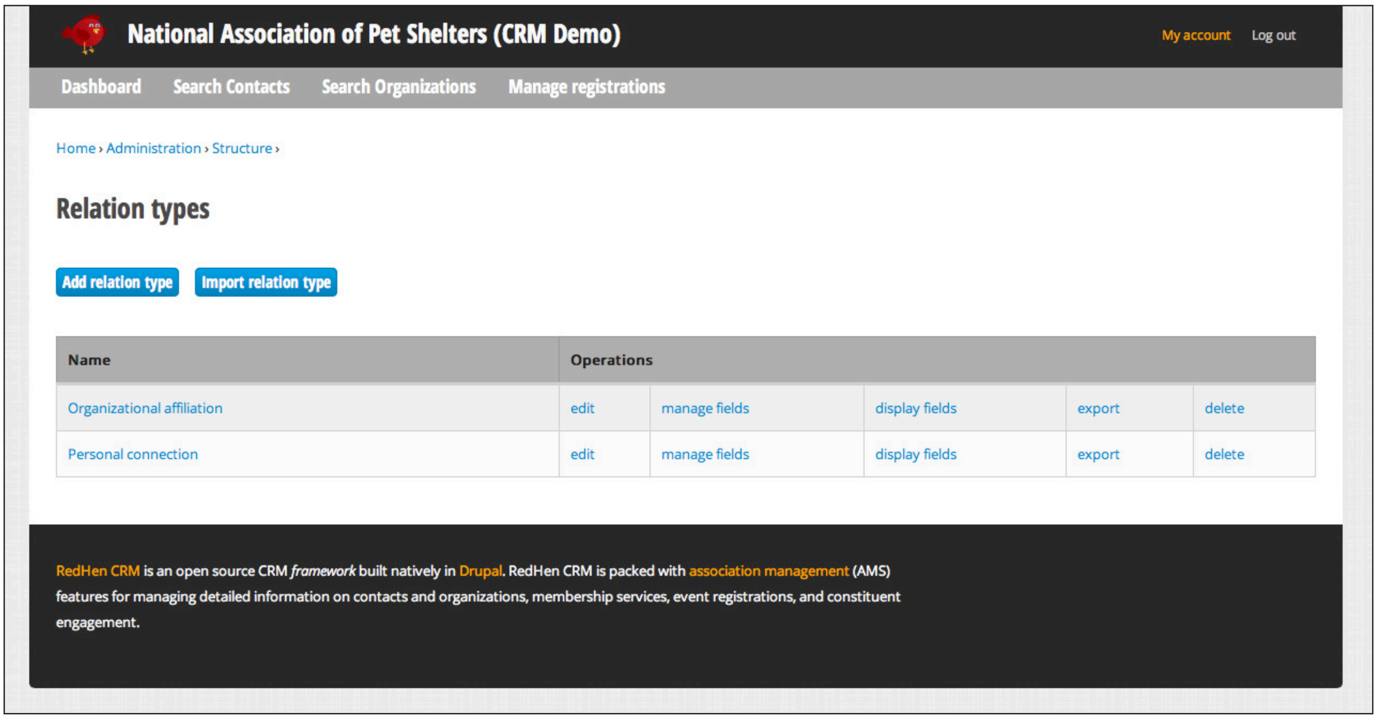
Figure 3. Custom Fields Being Exposed as Filters on a Listing of Contacts

- Memberships: memberships have an additional user role property that is inherited by linked Drupal users with a given membership. Create Premium and Standard memberships.

- Engagement Scores: engagement scores have an additional "score" property. Create High-value engagement and Standard-value engagement with arbitrary scores.

Now that your bundles have been created, you can add fields to them. Each bundle has a "manage fields" link. To manage fields for your staff contact type, for example, visit /admin/structure/redhen/contact_types/manage/staff/fields. You'll notice the e-mail field, which RedHen adds automatically behind the scenes to each contact bundle, because e-mail is required for several core components of the system. Incidentally, we developed our own e-mail field that mimics v-card structure allowing for multiple e-mail addresses per contact that have different labels and flags for default, bulk and on-hold statuses. In addition, the contact name is visible, although it doesn't have any settings to manage. This is because name is a property on a contact, which Drupal allows developers to expose as a "pseudo field", so that its position can be managed along with other traditional fields.

The interface for managing fields



Relation types

Name	Operations				
Organizational affiliation	edit	manage fields	display fields	export	delete
Personal connection	edit	manage fields	display fields	export	delete

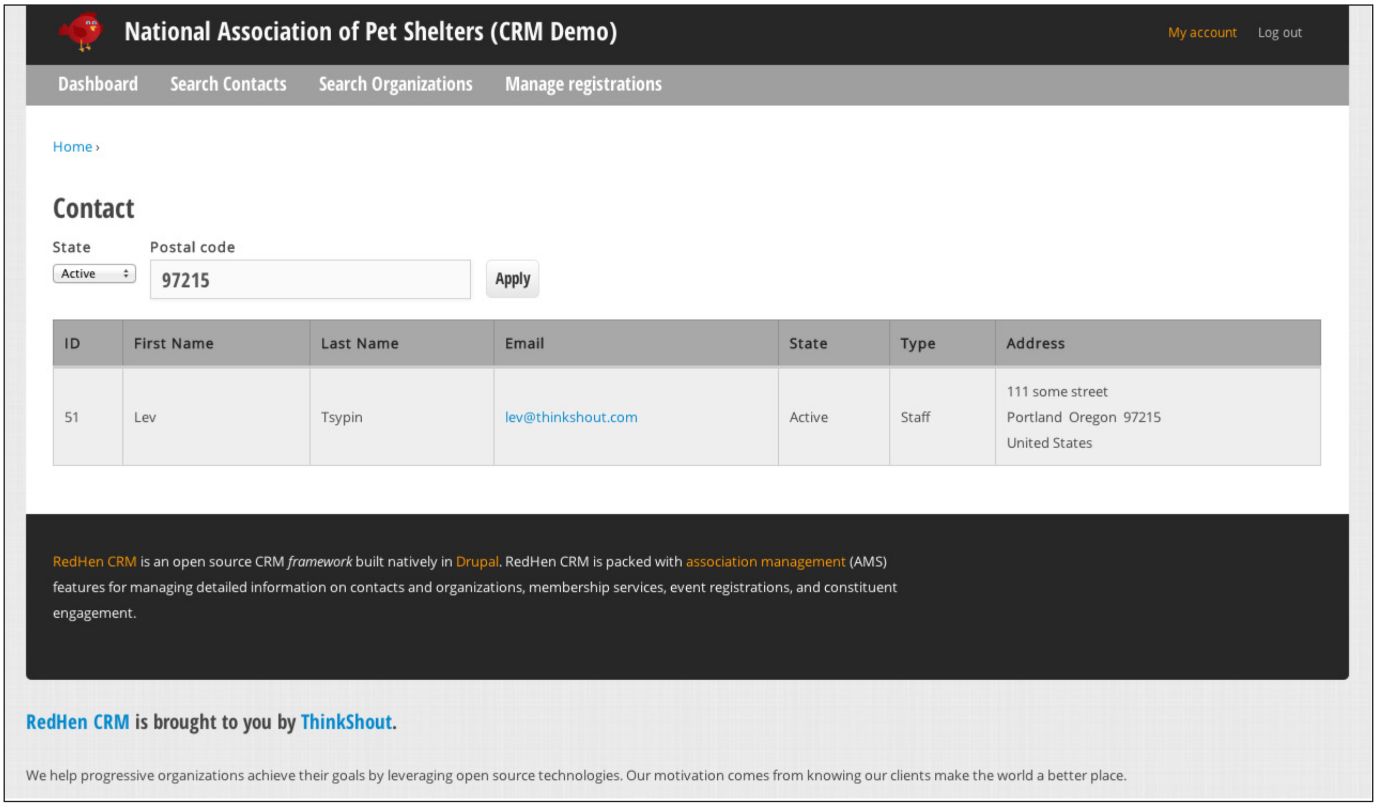
RedHen CRM is an open source CRM *framework* built natively in Drupal. RedHen CRM is packed with association management (AMS) features for managing detailed information on contacts and organizations, membership services, event registrations, and constituent engagement.

Figure 4. Listing of Default RedHen Relation Types

is part of Drupal core and is available when the Field UI module is enabled. Any field type can be added to your contact bundle, ranging from a simple text field to a complex address field, to an image or file upload field. Once a field is added to a bundle, it will be presented automatically when adding or editing an instance of the entity bundle (that is, an individual contact record), and it will be available to other components, such as Views or Rules. Fields attached to RedHen entities do not get added to the default tabular listings, but they are made available automatically as filters when changing bundles. You can override the default listings by creating a custom View with the same path and including any additional fields.

Relationships are another key component of RedHen and can connect contacts to organizations and other contacts. As mentioned earlier, RedHen installs two types of relations: Affiliations for connecting contacts to organizations and Personal Connections for relating contacts to each other. From the Relation types interface, you can add fields to these default relation types—for example, a position field to Affiliation—or create additional relation types, such as Employee.

When creating a relation, you must specify the types of source and


National Association of Pet Shelters (CRM Demo)

My account Log out

Dashboard Search Contacts Search Organizations Manage registrations

Home ›

Contact

State: Active

Postal code: 97215

Apply

ID	First Name	Last Name	Email	State	Type	Address
51	Lev	Tsypin	lev@thinkshout.com	Active	Staff	111 some street Portland Oregon 97215 United States

RedHen CRM is an open source CRM *framework* built natively in Drupal. RedHen CRM is packed with association management (AMS) features for managing detailed information on contacts and organizations, membership services, event registrations, and constituent engagement.

RedHen CRM is brought to you by ThinkShout.

We help progressive organizations achieve their goals by leveraging open source technologies. Our motivation comes from knowing our clients make the world a better place.

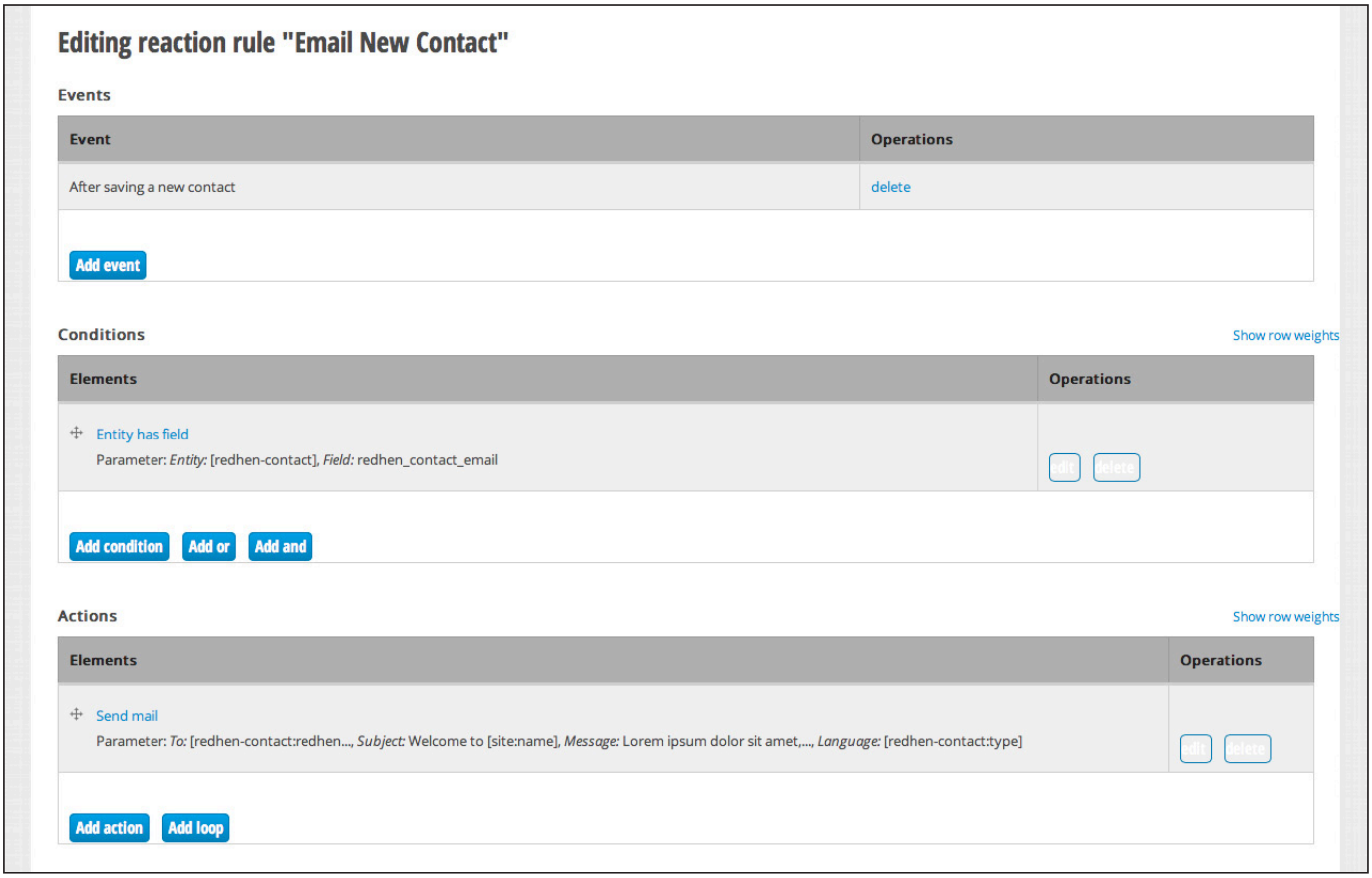
Figure 5. Custom Contact View with an Address and Exposed Filters

target entities and directionality of the relation. For the sake of this demo, let's stick with the default relation types.

Playing Well in the Drupal Sandbox

RedHen's default listings of contacts are a good start, but the fun really starts if you want to create your own custom views. The venerable Views module, a powerful graphical query builder, is the gold standard in Drupal for creating custom "lists of stuff", including RedHen entities. Much of the heavy lifting involved in Views integration is handled by the Entity API, but any module implementing the Entity API interfaces still needs to clarify the data types of all entity properties and provide Views handlers for any nonstandard data. For example, organization entities have a primary contact associated with them, and it's up to RedHen to explain to Views that the related contact ID within an organization record is actually a RedHen contact. Let's create a new View, adding additional fields and relationships to the standard list of contacts:

1. Create a new view at /admin/structure/views/add of Contacts.



Editing reaction rule "Email New Contact"

Events

Event	Operations
After saving a new contact	delete

Conditions

Elements	Operations
Entity has field Parameter: *Entity:* [redhen-contact], *Field:* redhen_contact_email	edit delete

Actions

Elements	Operations
Send mail Parameter: *To:* [redhen-contact:redhen..., *Subject:* Welcome to [site:name], *Message:* Lorem ipsum dolor sit amet,..., *Language:* [redhen-contact:type]	edit delete

Figure 6. Custom Rule That Demonstrates Sending an E-mail to New Contacts

2. Add relationships to Memberships and Organization affiliation.

3. Add the following fields: contact first and last name, contact e-mail (change the format to primary e-mail), membership name and organization name.

You now have a View of all contacts that includes their membership and organization. If you set the path of the View to /redhen/contact, it simply will override the default contacts listing, or you can move it elsewhere to maintain both interfaces.

Similarly, business rules can be extended using the Rules module, which provides an interface for defining logic based upon a trigger→action model. RedHen again leans on the Entity API to expose RedHen entities into this model, so that you can, for example, send an e-mail to a contact when a related membership entity is updated.

Extending Core Functionality

RedHen also can be extended and customized the old-fashioned way—by writing custom code. All of RedHen's entity types feature APIs with standard

CRUD operations and wrappers around many common tasks, such as getting all related entities. In addition, RedHen can be manipulated using Drupal's hook system. Each entity can be altered using a hook implementation when it's loaded, created, updated and deleted. Most hooks are exposed through the Entity API's inherited controller classes, but RedHen features some of its own unique hooks as explained in redhen.api.php. For example, a module can dictate whether a contact entity can be deleted by implementing hook_redhen_contact_can_delete():

```
function mymodule_redhen_contact_can_delete(RedhenContact $contact) {
  // prevent the deletion of active contacts
  if ($contact->redhen_state == REDHEN_STATE_ACTIVE) {
    return FALSE;
  }
}
```

In addition, all of RedHen's main interfaces are wrapped in theme functions so they can be overridden at the theme level. For example, to alter the default list of contacts, implement theme_redhen_contact_list():

```
function mytheme_redhen_contact_list($variables) {
  $contacts = $variables['contacts'];
  $header = $variables['header'];
  if (!empty($contacts)) {
    $rows = array();
    foreach ($contacts as $contact) {
      $uri = entity_uri('redhen_contact', $contact);
      $actions = array(
        l(t('edit'), $uri['path'] . '/view/edit',
        ➥array('query' => drupal_get_destination())),
        l(t('delete'), $uri['path'] . '/view/delete',
        ➥array('query' => drupal_get_destination())),
      );

      $redhen_contact_type = redhen_contact_type_load($contact->type);
      $rows[] = array(
        'data' => array(
          $redhen_contact_type->label,
          l($contact->first_name, $uri['path']),
          l($contact->last_name, $uri['path']),
          l($contact->email, 'mailto:' . $contact->email),
          format_date($contact->updated, 'short'),
          implode(' | ', $actions)
        )
      );
    }

    $render['table'] = array(
      '#theme' => 'table',
      '#header' => $header,
      '#rows' => $rows
    );
    $render['pager'] = array(
      '#theme' => 'pager',
    );
  }
  else {
    // no results, set a message
    $render['no-result'] = array(
      '#type' => 'markup',
        '#markup' => t('Sorry, there are no contacts
        ➥that match your criteria.'),
      );
    }

    return render($render);
  }
```

In short, using RedHen's comprehensive APIs, hooks and theme wrappers, a developer can build a complex, elegant solution to meet nearly any use case requiring CRM functionality.

Conclusion

Although we think it's possible to build very robust, large-scale CRM solutions natively in Drupal, ThinkShout's goal in releasing RedHen CRM is not to compete with the enterprise CRM market. Platforms like Salesforce almost inevitably will out-scale any CRM solution that we can build with Drupal.

That said, enterprise CRM solutions often are overkill for small to mid-size organizations and businesses. Moreover, even if your organization does need an enterprise CRM solution, we see RedHen as a natural integration point between your Web site and such a system. RedHen CRM opens the door to the creation of highly innovative front-end CRM tools. We anticipate that collecting and displaying data in RedHen (and Drupal) often will be much more affordable and nimble than trying to develop comparable features upon larger, more cumbersome enterprise packages.■

Sean Larkin has more than ten years of diverse leadership and technical consulting experience. He has been working with the best and brightest software engineers and designers in the Drupal community for the past five years. He has led national community organizing initiatives and international relief projects, served as a fundraising strategist for environmental groups worldwide, and ran two open-source software consultancies specializing in Drupal development. He holds a Master of Public Administration (MPA) degree from Syracuse University's Maxwell School. Outside work, Sean is an avid whitewater enthusiast, certified kayaking instructor, raft guide and whitewater videographer. He has been a longtime advocate and supporter of river conservation organizations, such as River Network and Waterkeeper Alliance.

Lev Tsypin has more than 12 years of experience leading technology projects and is a technical architect and co-owner of ThinkShout, Inc., where he leads technical design, user interface and module development. Prior to ThinkShout, Lev ran the highly respected consultancy, Level Online Strategy. In the 1990s, he worked as a consultant with Computer Sciences Corporation (CSC) and Inforte Corporation in Chicago. Before starting Level OS, he served as the Director of Programming at Pop Art, Inc. He holds a Bachelor's degree in Business Administration (BBA) and Political Science from the University of Wisconsin-Madison. When not working to help organizations through technology, Lev likes to take advantage of the areas they protect, be it on foot, skis or bike—at least when he's not wrapped up with his two boys, which isn't very often these days!

Resources

ThinkShout's Marketing Site for RedHen CRM: **http://redhencrm.com**

RedHen CRM Twitter Account: **http://twitter.com/redhen_crm**

RedHen CRM Source Code, Project Page and Issue Queue: **http://drupal.org/project/redhen**
(RedHen CRM download and usage statistics can be found here as well.)

The RedHen CRM Demonstration Drupal Installation Profile Download: **http://drupal.org/project/redhen_demo**

CiviCRM, the Open-Source CRM Solution: **http://civicrm.org**

Drupal Commerce, the Leading Drupal E-commerce Package with Which RedHen Integrates:
http://drupal.org/project/commerce

ThinkShout, Inc.: **http://thinkshout.com**

ThinkShout on Twitter: **http://twitter.com/thinkshout**

GOLD SPONSOR

New Relic:

A developer's best friend to monitor end-to-end performance of your PHP app.

http://www.newrelic.com

New Relic is the all-in-one web application performance management tool for the cloud and the datacenter. Its SaaS solution combines real user monitoring, application monitoring, server monitoring and availability monitoring in a single solution that has changed the way developers and operations teams manage web application performance in real time. More than 25,000 organizations use New Relic to optimize over 58 billion metrics in production each day. Fully implemented in just minutes, New Relic provides 24/7 real user monitoring and code level diagnostics for web apps deployed on dedicated infrastructures, the cloud or hybrid environments. New Relic provides support for PHP, Ruby, Python, Java, and .NET platforms and related frameworks, including Drupal. New Relic also partners with leading cloud management, platform and hosting vendors to provide their customers with instant visibility into the performance of deployed applications. New Relic partners include Acquia, ActiveState, cloudControl, Joyent, Pantheon, PHP Fog, and Tier 3.

New Relic is the only application performance management tool to offer enterprise-class capabilities—multi-language support, user monitoring, server monitoring, and deep-dive transaction analysis—in a single tool delivered as a service. Unlike traditional APM tools, it is easy to install, implement, and maintain, and requires no upfront investment. See for yourself why New Relic is indispensable for its customers. Or sign up today for a free 14-day trial of New Relic Pro.

p. 86, 87, 140, 141

SILVER SPONSOR

Acquia

https://www.acquia.com

Acquia empowers enterprises with the open source social publishing system Drupal. Co-founded by Drupal's creator in 2007, Acquia helps customers manage their growth and scale their online properties with confidence. Acquia's products, cloud infrastructure, and support enable companies to realize the full power of Drupal while minimizing risk, as it's done for more than 2,400 enterprise customers including Twitter, Warner Music Group, Turner Sports, World Economic Forum, Stanford University, Mercedes-Benz and NPR. See who's using Drupal at http://www.drupalshowcase.com, and for more information please visit www.acquia.com or call 888-9-ACQUIA.

p. 21

SILVER SPONSOR

Aten Design Group

http://atendesigngroup.com

aten design group

Aten Design Group is a web strategy, design, and development company based in Denver, Colorado.

We are Drupal experts and open source contributors. We design, build and support websites for organizations doing good work all around the world. We're passionate about meaningful user experience, beautiful design, and elegant open source solutions. Since 2000, we've worked with journalists, educators, human rights activists, health-care providers, climate change adaptation experts, international aid workers—a broad range of organizations committed to making positive changes in the world—to build interactive platforms that help impact lives.

We provide comprehensive strategy, design and development services to run all aspects of successful interactive projects, from start to finish. Our services include content strategy, information architecture, technical planning, design, development, support, dev-ops and server management. p. 43

BRONZE SPONSORS

Elevated Third http://www.elevatedthird.com

Founded in 2005, Elevated Third is a leading digital agency in Denver, Colorado. Through its unique combination of user-centered creative and open-source technology, Elevated Third is recognized for its digital marketing solutions that integrate intuitive user experience design using Drupal. The agency provides a variety of other award-winning services, including branding and usability and SEO consulting. For more information, visit elevatedthird.com, the company's Facebook page, or follow @elevatedthird. p. 51

FiberCloud, Inc. https://www.FiberCloud.com

FiberCloud is an IT and Cloud Infrastructure Provider. With a focus on superior service and support FiberCloud delivers colocation, cloud and connectivity solutions to businesses of all sizes. FiberCloud's infrastructure and cloud software solutions enable companies to securely and efficiently run their IT infrastructure and enable enterprise mobility, communication and collaboration. Our customized platforms allow developers to focus on their real work without worrying about the infrastructure. p. 121

ImageX Media http://www.imagexmedia.com

ImageX is an award-winning Drupal website design, development & consulting agency. Founded in 2001 as a full service media firm we've spent the past seven years refining our approach to building Drupal based sites. With a long history of contribution in the Drupal community and with over 100 successful Drupal site launches (including clients such as Disney, WB Records, Portland State University, Oncology Nursing Society, Vancouver School Board, Twit TV and AETN) we have the expertise needed to ensure success. p. 49

Advertiser Index

For more information on advertising in *Linux Journal* and special editions, please visit http://www.linuxjournal.com/advertising.

ADVERTISER	URL	PAGE #
Drupal Association	http://association.drupal.org/drupalcon http://DrupalSummit.com	23
Lullabot	http://www.lullabot.com	5

Speed Up Your Drupal Development Using Installations and Distributions

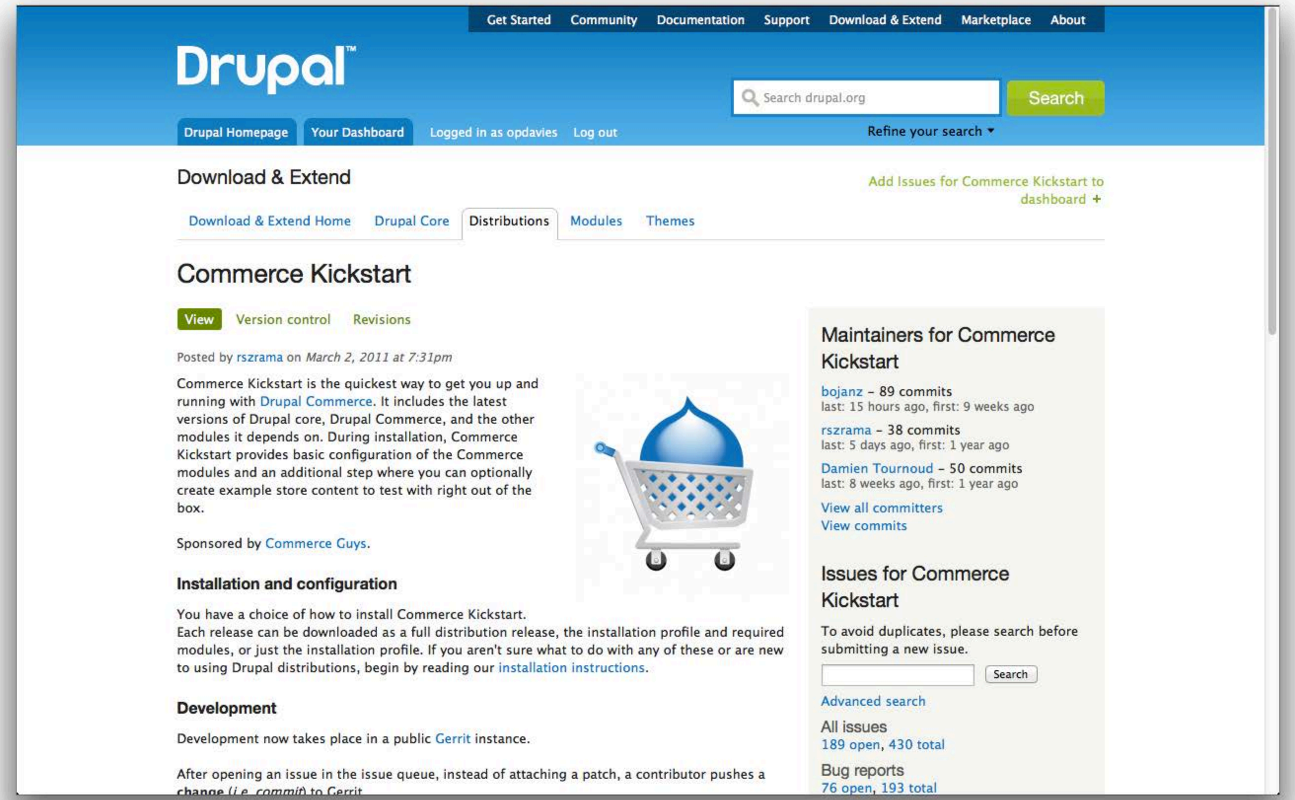
Create your own customized distribution to speed up your Drupal development process.

OLIVER DAVIES

Do you find yourself repeating the same steps whenever you start a new Drupal project? Do you always download and enable the same modules, and make the same configuration changes every time? As we start doing more and more Drupal projects at Nomensa, I noticed that we were doing exactly this, so I started to look into ways to streamline our initial project setup process. My solution was to create my own custom installation profile that provides me with a template to start each project, and this article outlines the steps I took to create it. The code outlined in this article has been committed into GitHub at **https://github.com/opdavies/linuxjournal_demo**, and it's available for you to download and re-use as needed.

What Are Installation Profiles?

Installation profiles are a combination of modules and themes and predefined configuration. A great example of an installation profile that I use regularly is called Commerce Kickstart (**http://drupal.org/project/commerce_kickstart**). It provides a version of Drupal 7 along with the Drupal Commerce suite of modules that



Download & Extend

Commerce Kickstart

Posted by rszrama on *March 2, 2011 at 7:31pm*

Commerce Kickstart is the quickest way to get you up and running with Drupal Commerce. It includes the latest versions of Drupal core, Drupal Commerce, and the other modules it depends on. During installation, Commerce Kickstart provides basic configuration of the Commerce modules and an additional step where you can optionally create example store content to test with right out of the box.

Sponsored by Commerce Guys.

Installation and configuration

You have a choice of how to install Commerce Kickstart. Each release can be downloaded as a full distribution release, the installation profile and required modules, or just the installation profile. If you aren't sure what to do with any of these or are new to using Drupal distributions, begin by reading our installation instructions.

Development

Development now takes place in a public Gerrit instance.

After opening an issue in the issue queue, instead of attaching a patch, a contributor pushes a change (*i.e. commit*) to Gerrit

Maintainers for Commerce Kickstart

bojanz – 89 commits
last: 15 hours ago, first: 9 weeks ago

rszrama – 38 commits
last: 5 days ago, first: 1 year ago

Damien Tournoud – 50 commits
last: 8 weeks ago, first: 1 year ago

View all committers
View commits

Issues for Commerce Kickstart

To avoid duplicates, please search before submitting a new issue.

All issues
189 open, 430 total

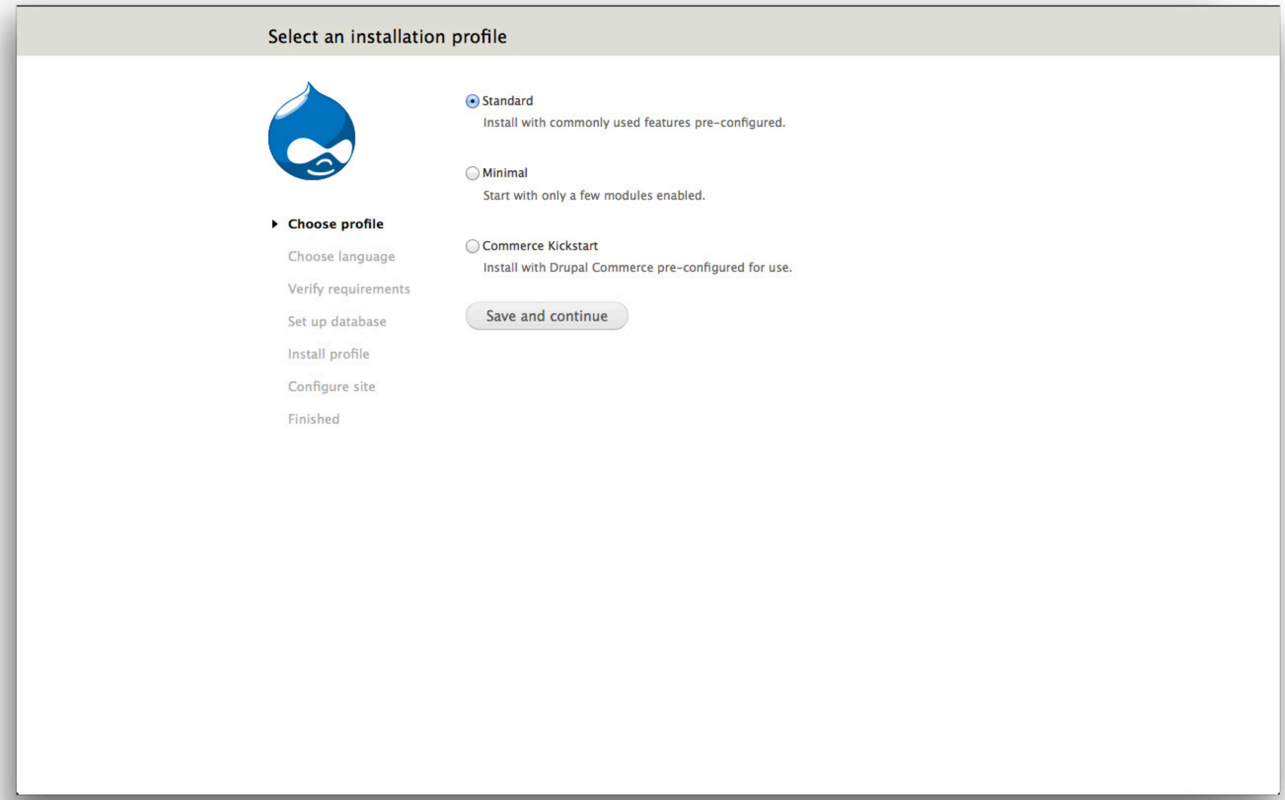
Bug reports
76 open, 193 total

Figure 1. The Commerce Kickstart Project Page

have been preconfigured to have the correct content types, rules, views and so on. Once an installation profile has been uploaded onto drupal.org as a project, it gets bundled with Drupal core, is available as a packaged download, and is known as a distribution. See **http://drupal.org/project/distributions** for a list of existing distributions.

How to Install an Existing Installation Profile

There are two different ways to download an existing Drupal installation profile. The first (and easiest) method is to download it as a distribution from **http://drupal.org**. To do this, go to the project page for the installation profile (for example, http://drupal.org/project/commerce_kickstart), scroll to the bottom of the page, and download a release in the same way that you would download a module or theme. The resulting file will be named something like commerce_kickstart-7.x-1.10-core.tar.gz, and this file will contain both Drupal core and the Commerce Kickstart installation profile. The other option is to download a fresh copy of Drupal core, and then download the installation profile seperately,


Select an installation profile

[✓ Standard]
Install with commonly used features pre-configured.

Minimal
Start with only a few modules enabled.

Commerce Kickstart
Install with Drupal Commerce pre-configured for use.

Figure 2. The Installation Screen with Commerce Kickstart

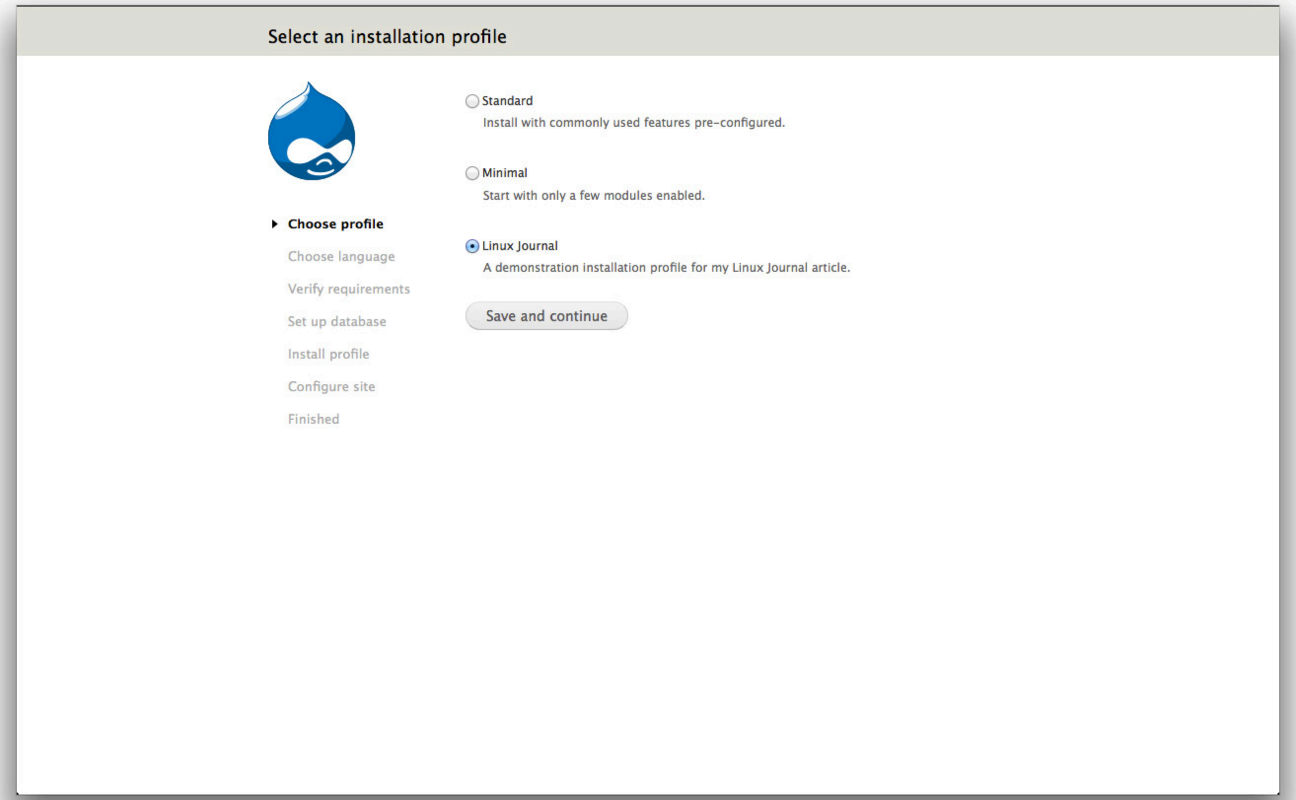
either using a drush command like `drush dl commerce_kickstart`, or clone it directly from its Git repository, and then place it within the Drupal's profiles directory.

Now, when you go to install Drupal, there is an additional option to use the Commerce Kickstart installation profile. Select the appropriate profile, click the Save and continue button, and continue through the installation process as normal.

How to Start Creating Your Own Installation Profile

Outside your Drupal directory, create a new directory to hold the files for your installation profile. Mine is called linuxjournal_profile, although the name of the profile is going to be simply linuxjournal. I've appended _profile to the end of the directory name. Later, there will be several different directories with similar names, so this helps provide some clarity. Be sure to give some thought to the name beforehand and ensure that it doesn't conflict with any potential modules or themes.

The first file I need to create is the linuxjournal.info file that, identical to .info files for modules and themes, defines the name and description of the profile, as well as which version of Drupal core that it is compatible with. I'm also


Select an installation profile

( ) Standard
Install with commonly used features pre-configured.

( ) Minimal
Start with only a few modules enabled.

(•) Linux Journal
A demonstration installation profile for my Linux Journal article.

Figure 3. The Install Screen with the Linux Journal Installation Profile

going to make it dependent on the core dblog and block modules so that these are enabled automatically when the profile is installed:

```
name = Linux Journal
description = A demonstration installation profile for my LJ article.
core = 7.x
dependencies[] = dblog
dependencies[] = block
```

The only other mandatory file needed for this profile to be visible to Drupal is the .profile file—in this case, linuxjournal.profile. Within this file, I can put any custom PHP functions or implementations of Drupal hooks that will take effect during the installation process. For now, I'm just going to add an opening PHP tag and leave the rest of the file blank. To confirm that everything works okay so far, I can download a fresh copy of Drupal core, copy linuxjournal_profile into the profiles directory, rename it to linuxjournal so that it is the same as the name of the profile, and load that site in a Web browser.

So far, so good. On the installation page, as well as seeing the default Standard and Minimal profiles, I also can see my Linux Journal profile. I can select this and continue with the installation process as normal to confirm that it's working, although because I've not entered anything into linuxjournal.profile

New Relic®

A DEVELOPER'S BEST FRIEND

See how we help 25,000+ customers monitor their apps.

Use the tool thousands of developers turn for immediate insight from the end user's behavior, through servers and down to the line of code. Performance management has never been so easy.

yet, nothing special is going to happen.

Adding a .install File

Identical to writing a module, I can create a .install file that contains functions to run when the profile is installed, updated or uninstalled. I'm going to utilize the hook_install() function from the minimal installation profile as part of my profile instead of re-declaring its contents myself and duplicating code. Within my own implementation of hook_install(), I can include the .install file from the minimal profile and then run its minimal_install() function:

```
<?php

/**
 * @file
 * Install, update and uninstall functions
 * for the Linux Journal installation profile.
 */

/**
 * Implements hook_install().
 *
 * Run the hook_install() function from the minimal
 * profile as part of this profile.
 */
function linuxjournal_install() {
  // Utilize the hook_install() implementation from
  // the minimal profile.
  include_once DRUPAL_ROOT . '/profiles/minimal/minimal.install';
  minimal_install();
}
```

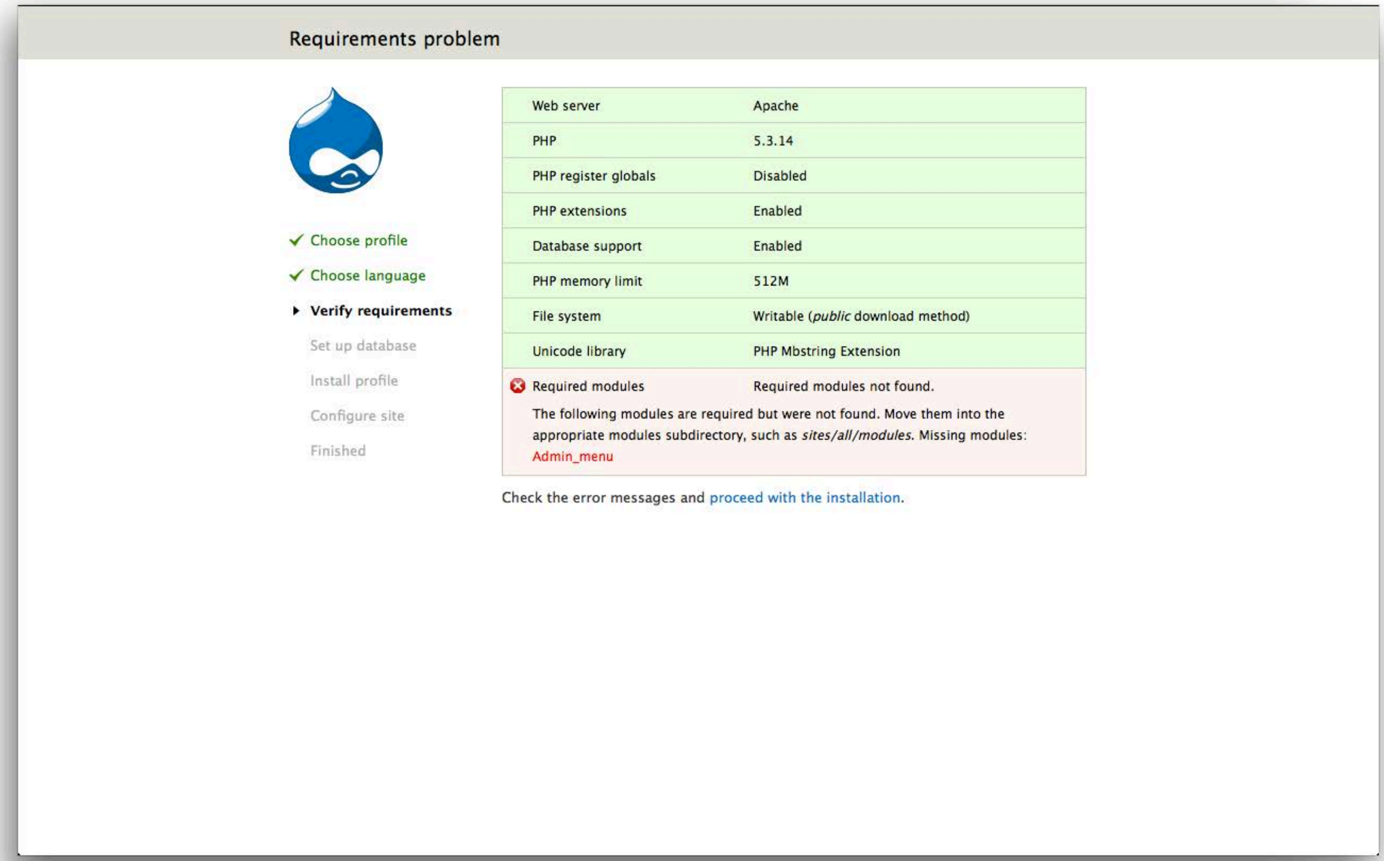
Adding Modules

I'm now going to create another file called linuxjournal.make. This file contains a listing of all the projects (modules and themes) and libraries that are used within the profile. My company specializes in building accessible Web sites, and I use a number of contributed modules on every site to help me do this. There are also some essential contributed modules, such as Administration Menu (**http://drupal.org/project/admin_menu**) and Pathauto (**http://drupal.org/project/pathauto**), as well as some custom modules and features that I use on every site. All of these will be listed within the linuxjournal.make file.

At the very top of linuxjournal.make, I need to add the following two lines to define the API version as well as the version of Drupal core that I'm using:

```
api = 2
core = 7.x
```

Here is the syntax for adding a project—in this case, the Administration Menu module—into a profile. Personally, I like to download projects from their Git repositories, although they could just be downloaded from drupal.org using wget. It's worth noting that the download URL doesn't have to be a URL at drupal.org. You can download projects from other sources, such as GitHub or BitBucket, or any other source, including from local


Requirements problem

- ✓ Choose profile
- ✓ Choose language
- Verify requirements
- Set up database
- Install profile
- Configure site
- Finished

Web server	Apache
PHP	5.3.14
PHP register globals	Disabled
PHP extensions	Enabled
Database support	Enabled
PHP memory limit	512M
File system	Writable (*public* download method)
Unicode library	PHP Mbstring Extension
Required modules	Required modules not found.

The following modules are required but were not found. Move them into the appropriate modules subdirectory, such as *sites/all/modules*. Missing modules: Admin_menu

Check the error messages and proceed with the installation.

Figure 4. The Error on Installation

file directories:

```
projects[admin_menu][type] = module
projects[admin_menu][subdir] = contrib
projects[admin_menu][version] = 3.0-rc3
projects[admin_menu][download][type] = git
projects[admin_menu][download][url] =
➥http://git.drupal.org/project/admin_menu.git
projects[admin_menu][download][branch] = 7.x-3.x
```

All modules specified will be located in the profiles/linuxjournal/modules directory by default; however, I prefer to store contributed modules in a subdirectory called contrib, hence the subdir value. I've also specified which version of the module to use and which branch within the Git repository to use. If I didn't specify a version number, the latest commit to the specified branch would be used.

To make the Administration Menu module enabled by default after installing the profile, I can declare it as a dependency by adding the following line into linuxjournal.info in the same way that I did for the dblog and block modules earlier:

```
dependencies[] = admin_menu
```

If I update my version of Drupal with these changes and try following the installation process, I will get an error because I've made the profile dependent on the Administration Menu module, but this module hasn't been downloaded yet for this instance of Drupal. The next step is to create a distribution that contains the linuxjournal installation profile, as well as all of the projects that are defined in linuxjournal.make.

Creating a Distribution

To create a distribution, the first thing I need to do is make a new file called distro.make that will be used to compile an instance of Drupal that contains the linuxjournal profile. As with linuxjournal.make, I need to start with declaring the API version and the version of Drupal core. I also need to declare Drupal core as a project as well as include the linuxjournal profile:

```
api = 2
core = 7.x

projects[drupal][type] = core
projects[drupal][version] = "7"

; Add the Linux Journal profile to the full distribution build.
projects[linuxjournal][type] = profile
projects[linuxjournal][download][type] = git
projects[linuxjournal][download][url] =
➥https://github.com/opdavies/linuxjournal_demo.git
```

Again, I'm using Git to download the installation profile and providing the distribution with the URL to the repository at GitHub. With distro.make saved, I now can compile Drupal using the following Drush command:

```
drush make distro.make directory
```

The last parameter is the name of the directory that you want Drupal to be compiled into. If one isn't specified, the directory that you're currently in will be used, which I wouldn't recommend. I want the resulting directory to be placed in the same level as my linuxjournal_profile directory, so I need to move up one level before specifying the name of the directory, which is going to be linuxjournal_demo:

```
drush make distro.make ../linuxjournal_demo
```

This command will download the latest version of Drupal 7 core, as well as all projects defined within linuxjournal.profile, into the linuxjournal_demo directory. Once that's compiled everything, within the profiles/linuxjournal directory, there is a new directory called modules/contrib that contains the admin_menu module. With all of the dependencies downloaded, I now can go to install.php again and go through the installation process without any errors.

Adding Themes into an Installation Profile

I can add a theme into my profile by declaring it within linuxjournal.make in the same way that I can for modules. As I use Omega as a base theme for most of my themes, I'll include that within my profile so that it's downloaded automatically:

```
; Themes ========================================================
projects[omega][type] = theme
projects[omega][version] = 3.1
projects[omega][download][type] = git
projects[omega][download][url] =
➥http://git.drupal.org/project/omega.git
projects[omega][download][branch] = 7.x-3.x
```

For now, I'll be creating the sub-theme manually, so I won't change the default theme here.

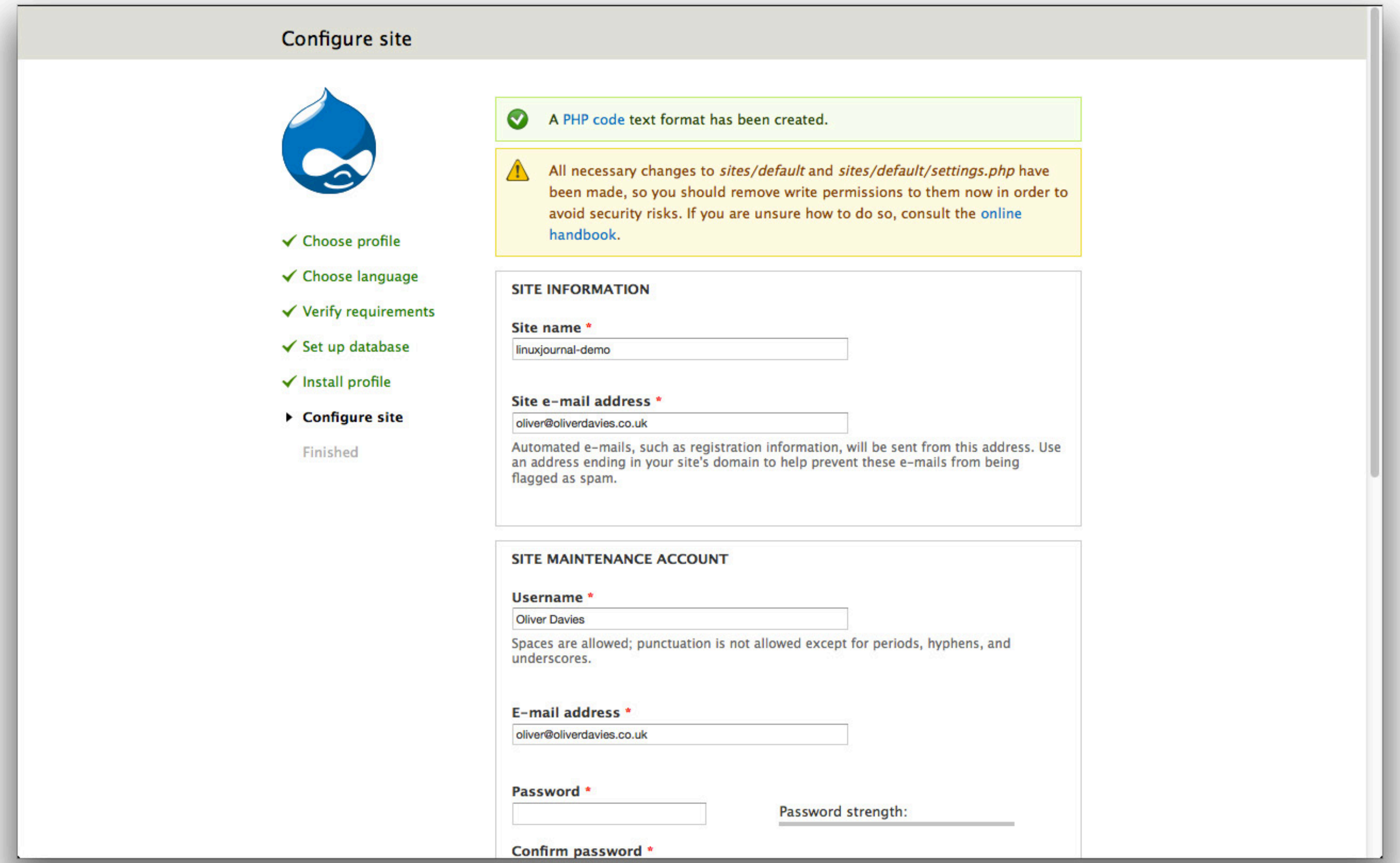
Customizing the Site Configuration Form

When going through the installation process, I usually make several changes to the site configuration form—namely setting a default site name and adding the default country. I can automate these changes by adding an implementation of hook_form_FORM_ID_alter() into linuxjournal.profile, as any functions defined in this file will take effect during the installation process. I know that the ID of the form is install_configure_form, and the name of the function that I'm going to create will be linuxjournal_form_install_configure_form_alter():

```
/**
 * Implements hook_form_alter().
 *
 * Allows the profile to alter the site configuration form.
 */
function linuxjournal_form_install_configure_form_alter(&$form,
➥$form_state) {
  // Set a default site name.
  $form['site_information']['site_name']['#default_value'] =
➥t('Linux Journal Demo');
}
```

As the form object is passed into the function by reference, I can add or overwrite information within the object by using this function. Here I'm setting a default value for the site's name field that will be pre-populated on the site configuration form the next time I follow the installation process. I can add and override any other values on the form by adding them into this function:

```
/**
 * Implements hook_form_alter().
 *
 * Allows the profile to alter the site configuration form.
 */
function linuxjournal_form_install_configure_form_alter(&$form,
➥$form_state) {
  // Set a default site name and email address.
  $form['site_information']['site_name']['#default_value']
```


Configure site

✓ Choose profile
✓ Choose language
✓ Verify requirements
✓ Set up database
✓ Install profile
▸ Configure site
Finished

A PHP code text format has been created.

All necessary changes to *sites/default* and *sites/default/settings.php* have been made, so you should remove write permissions to them now in order to avoid security risks. If you are unsure how to do so, consult the online handbook.

SITE INFORMATION

Site name *
linuxjournal-demo

Site e-mail address *
oliver@oliverdavies.co.uk
Automated e-mails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these e-mails from being flagged as spam.

SITE MAINTENANCE ACCOUNT

Username *
Oliver Davies
Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address *
oliver@oliverdavies.co.uk

Password *

Password strength:

Confirm password *

Figure 5. Site Configuration Form

```
➥= t('Linux Journal Demo');
  $form['site_information']['site_mail']['#default_value']
➥= 'linuxjournal@oliverdavies.co.uk';

  // Set a default username and email address.
  $form['admin_account']['account']['name']['#default_value']
➥= 'Oliver Davies';
  $form['admin_account']['account']['mail']['#default_value']
➥= 'linuxjournal@oliverdavies.co.uk';

  // Set a default country and timezone.
  $form['server_settings']['site_default_country']['#default_value']
➥= 'GB';
  $form['server_settings']['date_default_timezone']['#default_value']
➥= 'Europe/London';

  // Disable the 'receive email notifications' check box.
  $form['update_notifications']['update_status_module']
➥['#default_value'][1] = 0;
}
```

Now, as well as adding a default site name, I've also added a default site e-mail address, a default user name and e-mail address for the first user account, added a default time zone and country, and disabled the option to receive e-mail alerts when new updates are available.

For security reasons, I don't want to define my password in this file and will continue to enter this onto the form directly. I'm using #default_value and not #value so I can edit these predefined values on the form if I need to for this site. If I used #value, I would not be able to do so.

Setting the Administration Theme

Basically, what I now have is a copy of the minimal installation profile with some additional modules and themes. The first thing I want to do next is use the Seven theme for the administration pages of the site, as it would be if I'd used the standard installation profile. To do this, I first need to enable Seven and then set some variables to set it as the administration theme. I can reference the .install file from the standard installation profile to find out how to do this. To do so, I add the following code into the linuxjournal_install() function in linuxjournal.install:

```
// Enable the administration theme.
$admin_theme = 'seven';
db_update('system')
  ->fields(array('status' => 1))
  ->condition('type', 'theme')
  ->condition('name', $admin_theme)
  ->execute();
variable_set('admin_theme', $admin_theme);
variable_set('node_admin_theme', '1');
```

Creating Content Types

The standard installation profile also creates two default content types: basic pages and articles. I'm going to re-use most of this code to create the basic page content type for my profile:

```
// Add a 'Basic page' content type.
$types = array(
  array(
    'type' => 'page',
    'name' => st('Basic page'),
    'base' => 'node_content',
    'description' => st("Use <em>basic pages</em> for
➥your static content, such as an 'About us' page."),
    'custom' => 1,
    'modified' => 1,
    'locked' => 0,
  );
);

foreach ($types as $type) {
  $type = node_type_set_defaults($type);
  node_type_save($type);
  node_add_body_field($type);
}

// Default 'Basic page' to not be promoted and don't
// display author information.
variable_set('node_options_page', array('status'));
variable_set('node_submitted_page', FALSE);
```

To create additional content types, I can keep adding new items into the $types array, and they will be processed within the foreach() loop.

Adding Text Formats and a WYSIWYG Editor

When you use the standard installation profile, several different text formats are created. Currently, I have only plain text and PHP code (because the PHP module is now also a dependency for my profile). I also want the additional text formats, so I'll copy that section of code from the standard.install file and paste in into the linuxjournal_install() function:

```
// Add text formats.
$text_formats['filtered_html'] = array(
  'format' => 'filtered_html',
  'name' => 'Filtered HTML',
  'weight' => 0,
  'filters' => array(
    // URL filter.
    'filter_url' => array(
      'weight' => 0,
      'status' => 1,
    ),
    // HTML filter.
    'filter_html' => array(
      'weight' => 1,
      'status' => 1,
    ),
      // Line break filter.
    'filter_autop' => array(
      'weight' => 2,
      'status' => 1,
    ),
    // HTML corrector filter.
    'filter_htmlcorrector' => array(
      'weight' => 10,
      'status' => 1,
    ),
  ),
);
$text_formats['full_html'] = array(
  'format' => 'full_html',
  'name' => 'Full HTML',
  'weight' => 1,
  'filters' => array(
    // URL filter.
    'filter_url' => array(
      'weight' => 0,
      'status' => 1,
    ),
      // Line break filter.
    'filter_autop' => array(
      'weight' => 1,
      'status' => 1,
    ),
    // HTML corrector filter.
    'filter_htmlcorrector' => array(
      'weight' => 10,
      'status' => 1,
    ),
  ),
);
$text_formats['raw_html'] = array(
  'format' => 'raw_html',
  'name' => 'Raw HTML',
  'weight' => 2,
);
foreach ($text_formats as $text_format) {
  $text_format = (object) $text_format;
  filter_format_save($text_format);
}
```

I've amended the format slightly by creating an array called $text_formats, and then using a foreach loop to save each one as opposed to doing them separately as it is done in standard.install. I also usually create a Raw HTML format that allows all HTML tags and doesn't have any filters applied to it, so I've added it here as an additional format to be created automatically.

Adding TinyMCE and Enabling It for Certain Text Formats

I also download and install the WYSIWYG module and the TinyMCE editor for each site. Rather than having to download and enable the WYSIWYG module, download and extract the TinyMCE library and configure the editor, I can add it into my profile to have it done automatically. First, I need to add the WYSIWYG (**http://drupal.org/project/wysiwyg**) and Libraries API (**http://drupal.org/project/libraries**) modules and add them into dependencies to have them enabled by default.

In linuxjournal.make:

```
projects[wysiwyg][type] = module
projects[wysiwyg][subdir] = contrib
projects[wysiwyg][version] = 2.1
projects[wysiwyg][download][type] = git
projects[wysiwyg][download][url] =
➥http://git.drupal.org/project/wysiwyg.git
projects[wysiwyg][download][branch] = 7.x-2.x

projects[libraries][type] = module
projects[libraries][subdir] = contrib
projects[libraries][version] = 2.0
projects[libraries][download][type] = git
projects[libraries][download][url] =
➥http://git.drupal.org/project/libraries.git
projects[libraries][download][branch] = 7.x-2.x

libraries[tinymce][type] = library
libraries[tinymce][download][type] = get
libraries[tinymce][download][url] =
➥http://github.com/downloads/tinymce/tinymce/tinymce_3.5.6.zip
```

In linuxjournal.info:

```
dependencies[] = libraries
dependencies[] = wysiwyg
```

TinyMCE will be downloaded and extracted into the profiles/linuxjournal/profiles directory and will be accessible by the WYSIWYG module once Libraries API is enabled. However, now that it has been downloaded, I need to assign it to one of my text formats before it can be used. The format I'm going to assign it to is Filtered HTML, and I can do that by adding a record into the wysiwyg table in the database. I've also added an array of default settings that I've used on another site, and I've added this into the drupal_write_record() function:

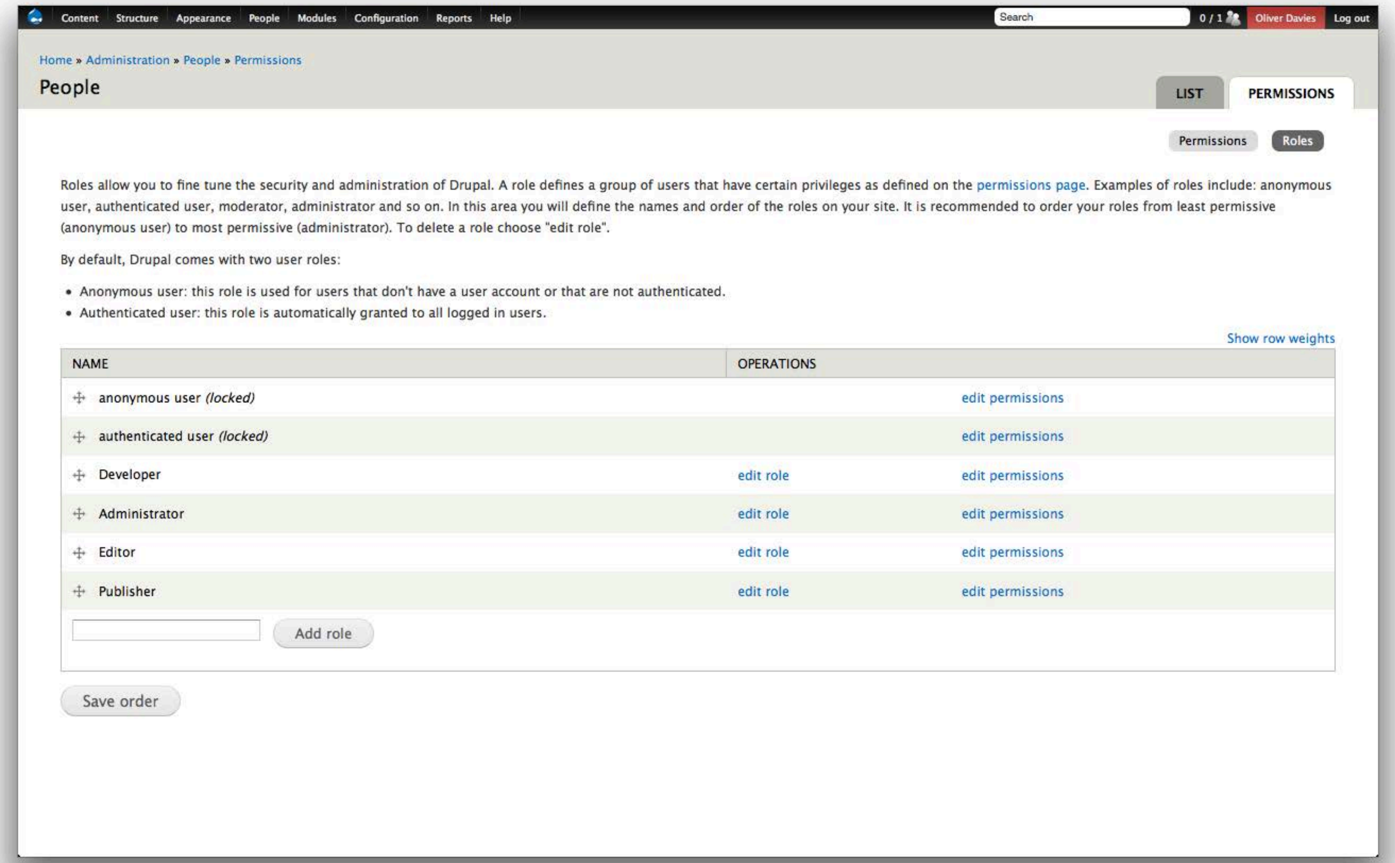
```
// Add the TinyMCE editor to the Filtered HTML text format.
```

```
$tinymce_settings = array(
  'default' => 1,
  'user_choose' => 0,
  'show_toggle' => 0,
  'theme' => 'advanced',
  'language' => 'en',
  'buttons' => array(
    'default' => array(
      'bold' => 1,
      'italic' => 1,
      'strikethrough' => 1,
      'justifyleft' => 1,
      'justifycenter' => 1,
      'justifyright' => 1,
      'justifyfull' => 1,
      'bulllist' => 1,
      'numlist' => 1,
      'link' => 1,
      'unlink' => 1,
      'anchor' => 1,
      'image' => 1,
      'formatselect' => 1,
      'sup' => 1,
      'sub' => 1,
      'blockquote' => 1,
      'code' => 1,
      'hr' => 1,
      'removeformat' => 1,
      'charmap' => 1,
    ),
  ),
  'toolbar_loc' => 'top',
  'toolbar_align' => 'left',
  'path_loc' => 'bottom',
  'resizing' => 1,
  'verify_html' => 1,
  'preformatted' => 0,
  'convert_fonts_to_spans' => 1,
  'remove_linebreaks' => 1,
  'apply_source_formatting' => 0,
  'paste_auto_cleanup_on_paste' => 1,
  'block_formats' => 'p,address,pre,h2,h3,h4,h5,h6,div',
  'css_setting' => 'theme',
  'css_path' => '',
  'css_classes' => '',
);

// Create the record.
$record = array(
  'format' => 'filtered_html',
  'editor' => 'tinymce',
  'settings' => $tinymce_settings,
);

// Save the record to the database.
drupal_write_record('wysiwyg', $record);
```

It's worth noting here that the $tinymce_settings variable needs to be a standard PHP array as it will be serialized automatically when it is written into the database table. When I first tried this, I tried importing a pre-serialized array into the database, which subsequently generated errors when trying to view the WYSIWYG admin settings form.

Creating Additional User Roles and Assigning Permissions

All of the Drupal sites that we build at my company have a moderation workflow in place provided by the Workbench



Roles allow you to fine tune the security and administration of Drupal. A role defines a group of users that have certain privileges as defined on the permissions page. Examples of roles include: anonymous user, authenticated user, moderator, administrator and so on. In this area you will define the names and order of the roles on your site. It is recommended to order your roles from least permissive (anonymous user) to most permissive (administrator). To delete a role choose "edit role".

By default, Drupal comes with two user roles:

- Anonymous user: this role is used for users that don't have a user account or that are not authenticated.
- Authenticated user: this role is automatically granted to all logged in users.

Show row weights

NAME	OPERATIONS	
anonymous user *(locked)*		edit permissions
authenticated user *(locked)*		edit permissions
Developer	edit role	edit permissions
Administrator	edit role	edit permissions
Editor	edit role	edit permissions
Publisher	edit role	edit permissions

Add role

Save order

Figure 6. The Additional Roles

Moderation module (**http://drupal.org/project/workbench_moderation**). This requires creating some additional roles for users who can edit and publish content. I also usually create a Developer role for use by any of the other Developers in the team that has permissions to all of the modules on the site, as well as an Administrator role for users who need more administrative access than Editors and Publishers, but who don't need full Developer-level access. Again, I can reference a section of code from standard.install and use that as a template:

```
// Create new user roles for Developers, Administrators,
// Editors and Publishers.
$roles = array('Developer', 'Administrator', 'Editor', 'Publisher');
foreach ($roles as $weight => $name) {
  $role = new stdClass;
  $role->name = $name;
  $role->weight = $weight + 2, // New roles must have at
                               // least a weight of 2.

  // Save the new role.
  user_role_save($role);

  if ($name == 'Developer') {
    // Give the Developer role all permissions.
    user_role_grant_permissions($role->rid,
➥array_keys(module_invoke_all('permission')));
```

```
    // Set this as the administrator role.
    variable_set('user_admin_role', $role->rid);

    // Assign user 1 the Developer role.
    db_insert('users_roles')
      ->fields(array('uid' => 1, 'rid' => $role->rid))
      ->execute();
}
```

I've created an array containing the names of the new roles I want to create, and then a foreach() loop including the key of each item that will be used to define the weight for each role. The first section applies to all new roles where the role is created, and then there is an additional section that applies only to the Developer role. This section assigns all permissions to the Developer role as well as assigning the role to user 1.

With the new roles created, I now can set some default permissions:

```
// Assign some default permissions.
$filtered_html_permission =
➥filter_permission_name($filtered_html_format);
$raw_html_permission = filter_permission_name($raw_html_format);
user_role_grant_permissions(DRUPAL_ANONYMOUS_RID,
➥array('access content', $filtered_html_permission));
user_role_grant_permissions(DRUPAL_AUTHENTICATED_RID,
➥array('access content', 'access administration menu',
➥'access devel information', $filtered_html_permission,
➥$raw_html_permission));
```

To start, I find out the name of the permissions for the Filtered HTML and Raw HTML text formats that I created earlier. Anonymous users are going to have very restricted permissions and are only going to be able to access content and use the Filtered HTML permission. The authenticated users also will be able to access the Administration Menu and information from the Devel module, as well as the Filtered HTML and Raw HTML text formats.

These are just a few examples of what can be done using installation profiles and distributions, and I've shown some of the ways I automate the Drupal installation and configuration processes. The complete installation profile I'm currently using at Nomensa also includes custom themes with theme template overrides that then can be used on custom public-facing and administration themes and additional Drupal configuration. This has saved me hours of time in the initial development stages as opposed to having to redo the same steps each time I start developing a new site.■

Oliver Davies is an Application Developer at Nomensa, specializing in Drupal and PHP development. Outside work, Oliver is a keen contributor to the Drupal community—attending local user groups, maintaining several of his own contributed modules, and blogging about the latest Drupal developments and issues.

Trekk—a Drupal Distribution for Universities

Take a close look at how Trekk uses Flatfish to scrape and share content.

TIM LOUDON

Trekk is a new, innovative Drupal distribution focusing on higher education. Trekk's primary goals are to enable flexible and intelligent content sharing and to simplify launching on to Drupal. As one of Trekk's main developers, I definitely can say that Trekk just wants to make your life easier.

A couple outstanding Drupal distributions already exist (Open Scholar and Open Academy), so I want to outline some use cases to help illustrate Trekk's niche. Primarily, Trekk assumes you either have or want to have several Drupal sites that display the same content. For example, the PR team wants important news and events to show up on every school's or department's Web site, or several professors are cross-listed on department or school sites, or a certificate program contains course information spread across several departments. Traditionally, duplicate content is either (tediously) maintained, out of date or simply but sub-optimally omitted. Trekk aims to solve these key problems for you. Given that universities often have dozens of key Web sites and thousands of pages of content, the other half of Trekk focuses on migration onto Drupal.

If you've ever dealt with homegrown PHP sites, Oracle's dismal 64-bit Linux support, dual-booting Windows to run Microsoft SQL Server or just a whole mess of static HTML pages, you know why I wrote Flatfish (if not, take my word that there was sufficient motivation). Flatfish is a Ruby gem that abstracts the legacy platform's technology stack, so you can use CSS selectors à la jQuery to scrape only the content you want. Trekk then provides built-in support to clean and migrate this data into Drupal.

When I talk about sharing content across sites, I really mean more than that. If you just wanted to display some simple content on two or more sites, you could do that using Drupal core and standard contrib. The Views module could provide an outbound RSS feed, and the other site(s) could use Feeds to consume the content. This architecture falls down though when you have complex data and/or what I would call a real-world workflow. Trekk conceptually builds on this Views+Feeds approach to create a lightweight content server. A Trekk server uses Views and Services to expose JSON endpoints. Trekk clients then can access the server securely and get the content they need.

Additionally, Trekk clients can manipulate the data into multiple formats. For example, a client site can consume the JSON and create Drupal nodes. This lets the client site control the display through the use of Views, Display Suite or Panels. You also could write a simple extension to the Trekk Server to share HTML, and then client sites could show this directly via Panels. It allows for a tremendous amount of flexibility in how the client sites use the shared content. This is key in helping school and department sites maintain their autonomy, but it also allows for better brand compliance and centralized control where appropriate. Trekk allows you to build a system that meets your unique requirements. The Trekk client is robust and uses Migrate under the hood to create the Drupal nodes. There is the added benefit of controlled updates and access to the Migrate plugin ecosystem. And, returning to the Views+Feeds comparison, Trekk preserves relationships. If there's a Faculty content type and a Publication content type, the connections between the nodes on the server are passed on to the clients.

As for Flatfish, you're probably wondering, "Why would a PHP Drupal developer create a tool in Ruby?" When I initially designed Flatfish, I knew that I wanted something flexible, dynamic and, honestly, a bit magical (but hopefully not cryptic). Ruby seemed like an ideal choice, but two libraries made it the only one: Nokogiri and Active Record. Nokogiri is an awesomely cool Ruby gem that provides CSS selectors as well as XPath for XHTML and XML. The end result is fine-grained control over what HTML gets scraped. This has worked out surprisingly well, even on pure HTML sites where there aren't CMS templates to force all the desired content into the div#main or div#content. The other library, Active Record, is one of Ruby on Rails' core libraries. It provides an ORM, so all persistent data manipulation in Flatfish is done via Objects. This has been incredibly handy, as Flatfish creates database tables on the fly and adapts those tables to mirror the

Table 1. Contents of bio.csv

URL	Path	Body	field_position
http://drupalconnect.com/team/john-florez	bio/john-florez	John Florez	.field-name-field-user-bio .field-name-field-user-position .field-item
http://drupalconnect.com/team/jonathon-whitener	bio/jonathon-whitener	Jonathan Whitener	.field-name-field-user-bio .field-name-field-user-position .field-item
http://drupalconnect.com/team/tim-loudon	bio/tim-loudon	Tim Loudon	.field-name-field-user-bio .field-name-field-user-position .field-item
http://drupalconnect.com/team/mike-crittenden	bio/mike-crittenden	Mike Crittenden	.field-name-field-user-bio .field-name-field-user-position .field-item
http://drupalconnect.com/team/christopher-jones	bio/christopher-jones	Christopher Jones	.field-name-field-user-bio .field-name-field-user-position .field-item

destination Drupal system's content types transparently. Outside of Active Record and Nokogiri, Flatfish handles the mundane migration details—normalizing links and tokenizing and saving media as binary blobs.

Now that you know a little about Trekk and Flatfish, let's get our neckbeards on and walk through an example. For Flatfish, download Ruby 1.9.3, and install all of the soft dependencies (also save yourself some time and install libxml2-dev, libxslt-dev and mysqlclient-dev or your distro's equivalents—gem dependencies). A note on the soft dependencies, Ruby will install without them, but you don't want that so be sure to grab them, in particular libyaml. Rubygems, your PHP PEAR/PECL equivalent, comes with Ruby, so now you can install the Flatfish gem. This usually means typing `sudo gem install flatfish`. If you're struggling here, any guide on installing Rails should help, or you can try using rbenv+ruby-build or RVM to assist with the installation and management of Ruby versions, implementations and gemsets. However, be aware that rbenv and RVM come with their own set of concerns and are recommended for those interested in Ruby beyond Flatfish. Finally, there's also an IRC channel: #trekk. For gem version incompatibilities, use the Flatfish GitHub Issues.

Using Flatfish is straightforward. The program parses a CSV. Each row contains the URL to be scraped, metadata about the URL for Drupal and one or more columns of CSS selectors. Each CSS selector maps to a Drupal field.

For an example, let's scrape the Drupal

Connect Web site and populate a Bios Content Type as well as a handful of blog posts to demonstrate the media handling. The first step is to create the CSVs. The bio CSV contains a header row with URL, title, path, body and an extra field for the employee's position or title within the company. Let's use the machine name for the position field, field_position. The blog posts will correspond to Drupal Articles and don't have any extra fields.

The full bios CSV as well as the articles CSV are available in the Flatfish repo on GitHub. You can see that the CSS classes match the DC site's bio and position. For each URL, you eventually will have a Drupal node. The HTML from the ".field-name-field-user-bio" selector will go in the Drupal node body, and HTML from the ".field-name-field-user-position .field-item" will go in the node position field. The aforementioned metadata includes the Drupal node URL alias and node title. Note that Flatfish dynamically creates table schemas, so you can add as many CSV columns after "body" as you need. And although it's not used here, there's concatenation support. This comes in handy if you need to skip an image or menu in the middle of your target URL's content. For example, if a CSS selector field's value is "#main-content p.one && #main-content p.three", Flatfish would concatenate the text of those two CSS selectors ostensibly skipping HTML in the "#main-content p.two" tag.

Now copy the YAML file below and change the values to match your specifics:

```
# DB credentials
db_user: 'root'
db_pass: 'correcthorsebatterystaple'
db: 'flatfish_sample'

# NOTE: these map to Drupal content types
# and AR database tables
types:
  Bio:
    csv: '/home/tloudon/lj/example/bio.csv'
    host: 'http://drupalconnect.com'
  Article:
    csv: '/home/tloudon/lj/example/article.csv'
    host: 'http://drupalconnect.com'

# use the web, otherwise path to local HTML root
local_source: ''
```

As implied, Flatfish supports scraping multiple Content Types in a single session; just add them into the file as shown below or see the GitHub repo's example directory. Also, don't forget to create the database. Subsequent migrations can use the same database and will update the content, keeping the same IDs. And, that's it; setup is complete.

Flatfish supplies a binary (remember to `exec $SHELL` to update your $PATH right after the gem install), so run `flatfish` in a terminal in the directory with config.yml. During runtime, Flatfish will pull down the content over the Net and update all of the relative links. (There also is support for a local importation if you have a copy of the site or use wget to mirror it, which can speed up the migration substantially.) Flatfish will create two or more database tables: a media table and one for each Content Type. Any files (images, PDFs, docs and so on) referenced in the CSS selected HTML are saved in the media table. Flatfish considers URLs unique, so only one copy of an image or file is saved even if there are multiple references to it. Note that the scraped HTML now has tokens to the media table. Take a moment to query the database and verify that all content saved correctly. Again, Flatfish supports updates and maintains IDs, so you can rerun Flatfish several times during a session or during the course of a project.

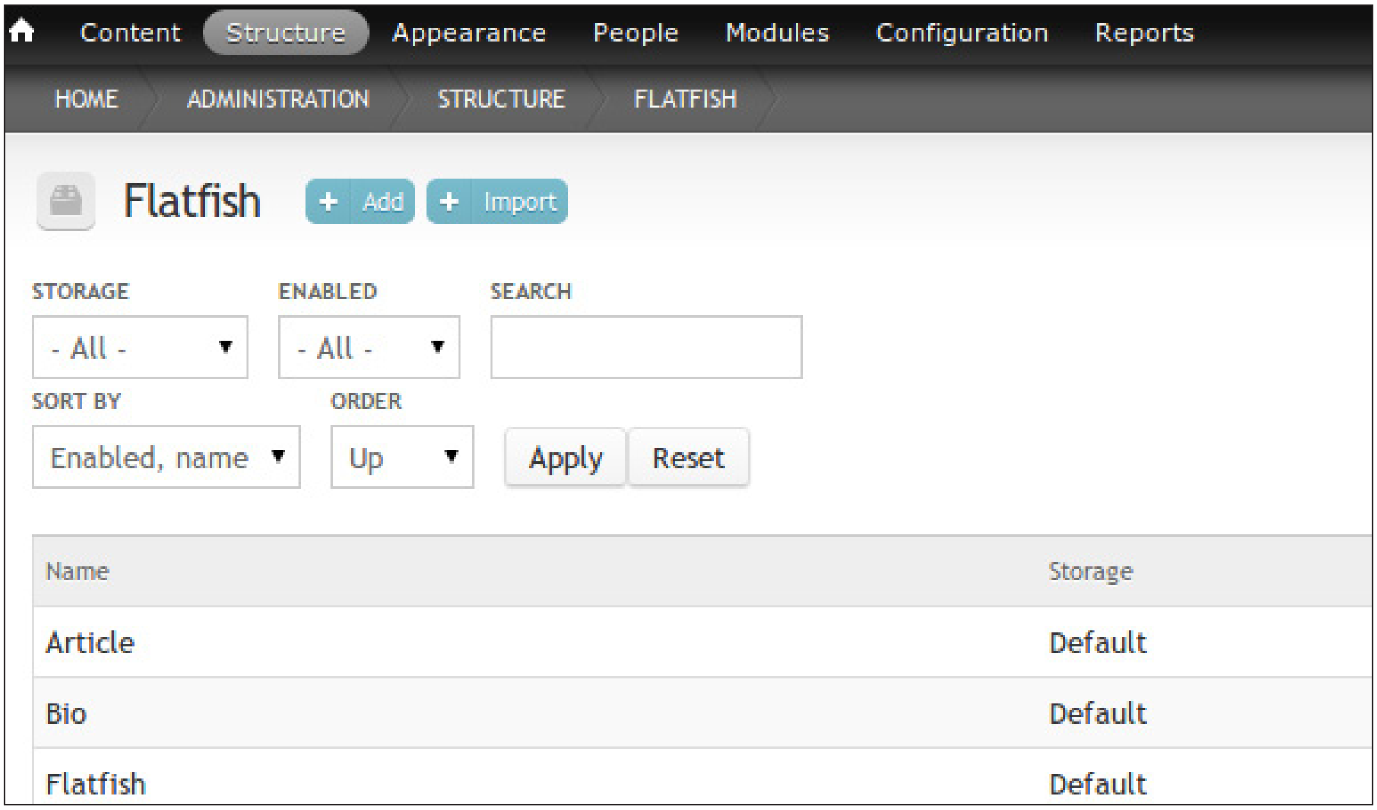
On the Drupal side, we've created a Flatfish module that's included in Trekk or available on drupal.org. Trekk has a drush makefile to download Flatfish's required libraries; however, you also can download them manually (queryPath, HTMLPurifier and Spyc). Spyc provides YAML support; queryPath replaces the media tokens, and HTMLPurifier strips out all nonsemantic markup. The heart of the module uses Ctools plugins and Migrate to provide a reasonably automated, dynamic migration.

Download the Trekk tarball, run the drush makefile as specified in the README.md, and choose a "Trekk Server" with "Flatfish Support".

The Flatfish module needs access to the config.yml, the Flatfish generated schema.yml and the scraped HTML database. Create a new directory called flatfish_migrations in sites/all, and place the YAML files in it. Review the generated schema.yml file, and ensure that each Content Type's "machine_name" matches the Drupal Content Type machine name. (Note: Rails pluralizes database tables, so a Flatfish "Bio" type has a "bios" table and matches a "bios" Drupal content type.) Navigate to admin/structure/flatfish. The Flatfish module parses the YAML files and dynamically creates the configuration page items. When you visit the Flatfish module configuration page, the items are registered as Migrate classes. This means database tables are created to store important Migration data, and you now can see and run the Migrations under admin/content/migrate.

As implied above, all media, regardless of content type, is stored in the same database table. Run the media Migration first; this will create the physical files and add them to the file_managed database table. The files are now accessible under admin/content/media and are no different from any other natively created Drupal files.

Next, run all Node/HTML Migrations.

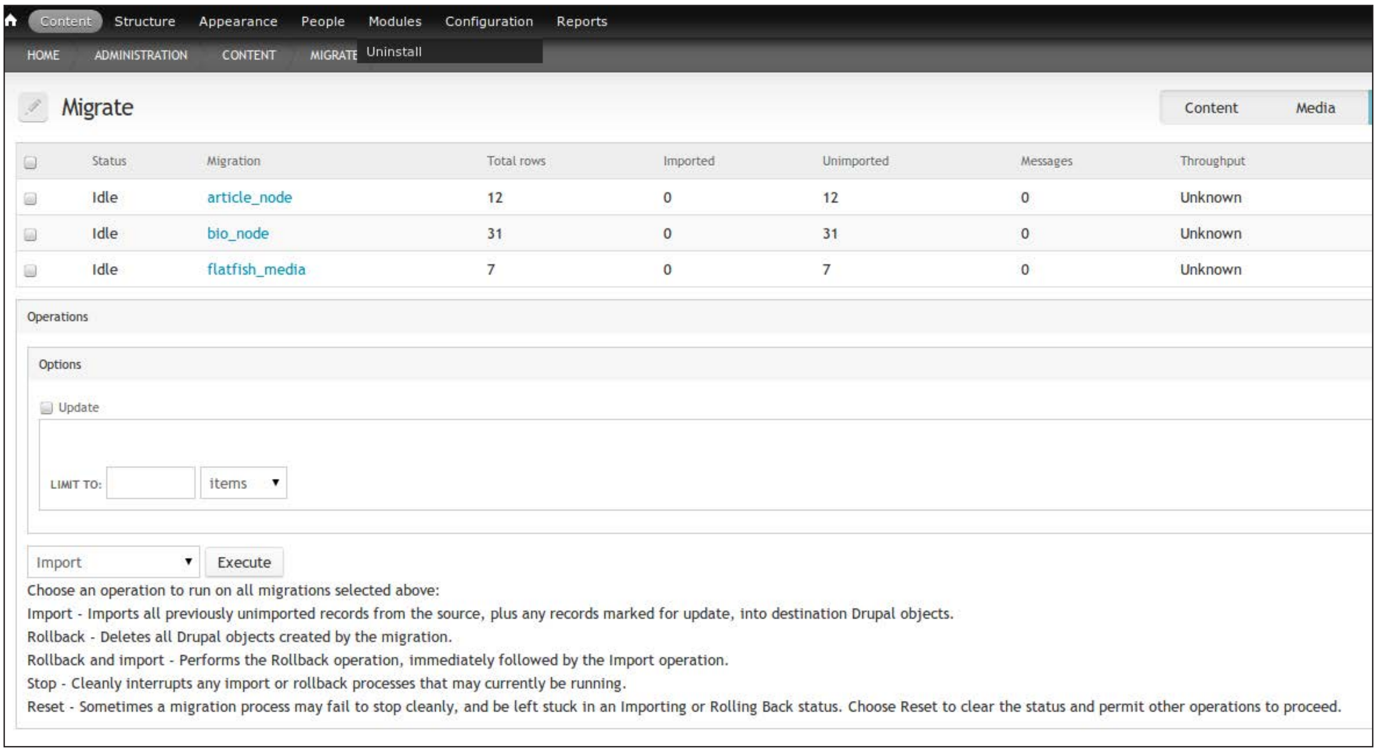


Flatfish

STORAGE: - All -
ENABLED: - All -
SEARCH:
SORT BY: Enabled, name
ORDER: Up

Name	Storage
Article	Default
Bio	Default
Flatfish	Default

Figure 1. Flatfish Module Configuration Page



Migrate

	Status	Migration	Total rows	Imported	Unimported	Messages	Throughput
☐	Idle	article_node	12	0	12	0	Unknown
☐	Idle	bio_node	31	0	31	0	Unknown
☐	Idle	flatfish_media	7	0	7	0	Unknown

Operations

Options

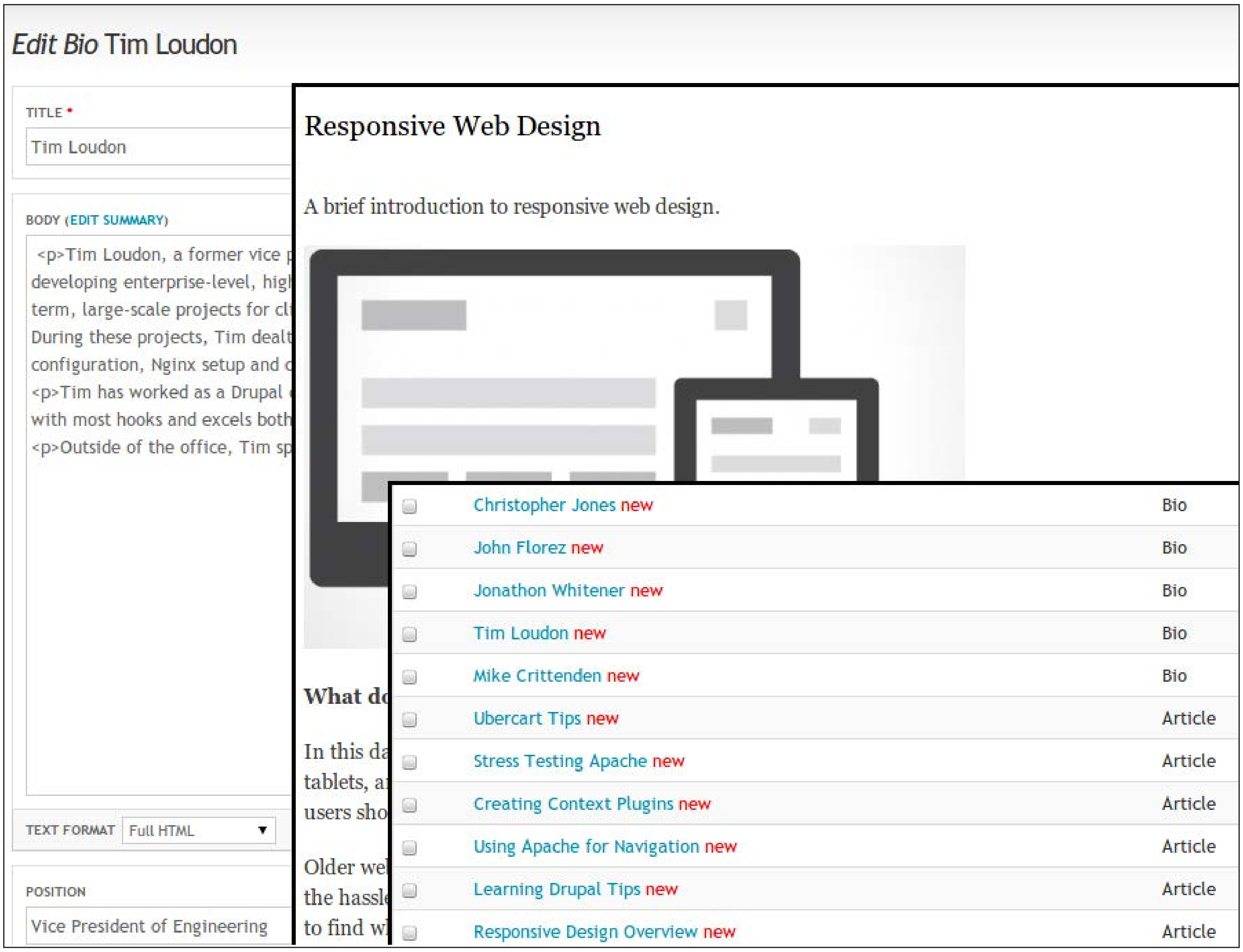
☐ Update

LIMIT TO: items

Import

Choose an operation to run on all migrations selected above:
Import - Imports all previously unimported records from the source, plus any records marked for update, into destination Drupal objects.
Rollback - Deletes all Drupal objects created by the migration.
Rollback and import - Performs the Rollback operation, immediately followed by the Import operation.
Stop - Cleanly interrupts any import or rollback processes that may currently be running.
Reset - Sometimes a migration process may fail to stop cleanly, and be left stuck in an Importing or Rolling Back status. Choose Reset to clear the status and permit other operations to proceed.

Figure 2. Migrate UI


Edit Bio Tim Loudon

TITLE *
Tim Loudon

BODY (EDIT SUMMARY)
<p>Tim Loudon, a former vice p
developing enterprise-level, high
term, large-scale projects for cl
During these projects, Tim dealt
configuration, Nginx setup and c
<p>Tim has worked as a Drupal
with most hooks and excels both
<p>Outside of the office, Tim sp

TEXT FORMAT Full HTML

POSITION
Vice President of Engineering

Responsive Web Design

A brief introduction to responsive web design.

What do

In this da
tablets, a
users sho

Older web
the hassle
to find wh

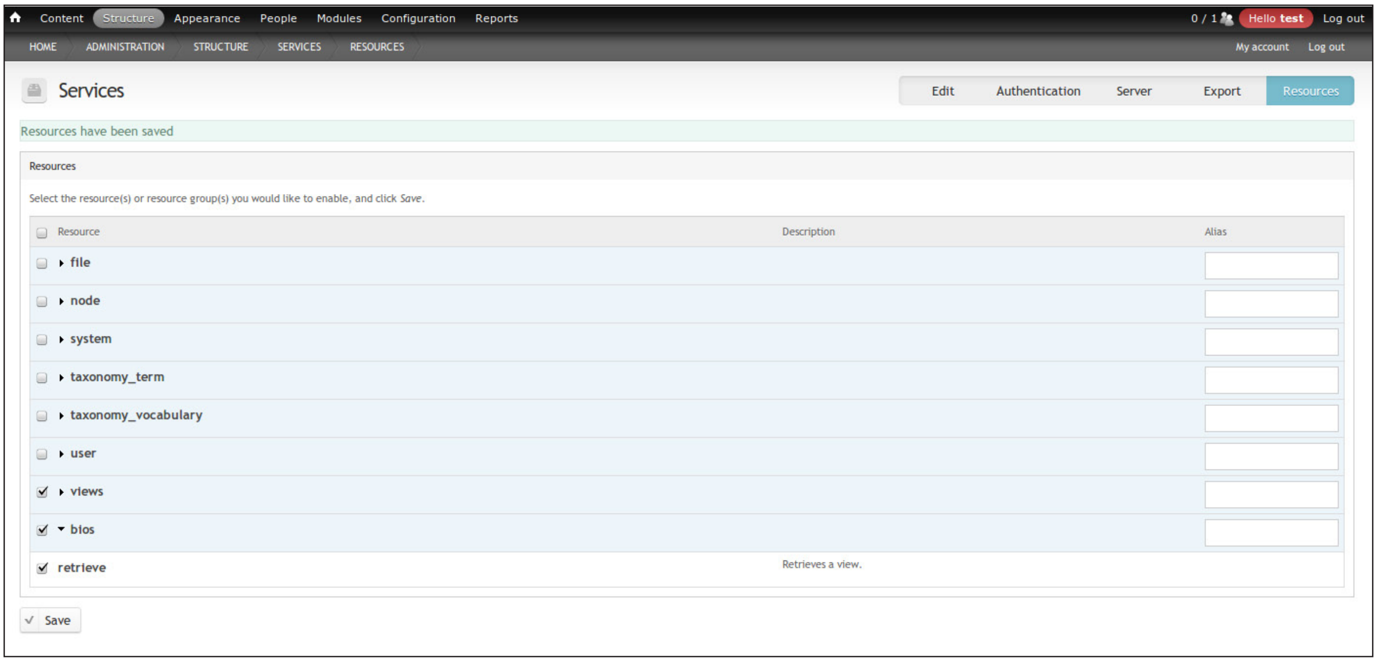
Christopher Jones new	Bio
John Florez new	Bio
Jonathon Whitener new	Bio
Tim Loudon new	Bio
Mike Crittenden new	Bio
Ubercart Tips new	Article
Stress Testing Apache new	Article
Creating Context Plugins new	Article
Using Apache for Navigation new	Article
Learning Drupal Tips new	Article
Responsive Design Overview new	Article

Figure 3. Example migrated content: node/edit sample showing migrated title, body and position; sample image in a responsive design blog post; and a thumbnail of admin/content.

As mentioned previously, each URL from the CSV corresponds to a Drupal node. Because the HTML has simple Flatfish media tokens, it's important that you run the media Migration first, so that the files exist in Drupal and the system will update them. If you skipped the media first, click the update check box, and run the Node/HTML Migrations again. You also will notice that the HTML is clean. HTMLPurifier accepts a comma-delimited whitelist of HTML tags with attributes. Flatfish has defaulted to stripping all divs, spans, classes and ids. Although this may change if it proves to be controversial, the Flatfish module strips HTML table tags as well. In my experience, legacy HTML tends toward table-driven layout rather than legitimate tabular data presentation. If you need to import tables, either hack the module (and kill a kitten) and add the three HTML table tags to the whitelist, or file an issue on drupal.org, so we can see if there is a community need for a configurable whitelist. The major advantage of the HTML stripping is that it allows the Drupal theme to define the layout without worrying about or writing overrides for the legacy HTML. However, there also can be a huge compliance benefit in that any insecure HTML (inputs, forms), JavaScript or iframes also are removed.

Now that the example content is in Drupal, let's look at sharing it. Trekk's server code is fairly lightweight and builds upon several other Drupal modules. We have written a custom Views handler that outputs JSON. And, Services provides a RESTful endpoint that client servers can contact securely.

Next create a View. For this example, let's create one for the Bios Content Type. This is a normal View and can use filters, contexts, relationships and so on. Now configure your Services Server. We recommend the REST server. At the time of this writing, you will need to symlink the Spyc PHP file that Trekk's drush makefile installs in sites/all/libraries to the rest_server/lib directory. Check your endpoint configuration, and enable the View as a "retrieve" resource. Services Servers can set authentication and have configurable paths. The example endpoint is available at



Resources have been saved

Resources

Select the resource(s) or resource group(s) you would like to enable, and click *Save*.

Resource	Description	Alias
☐ ▸ file		
☐ ▸ node		
☐ ▸ system		
☐ ▸ taxonomy_term		
☐ ▸ taxonomy_vocabulary		
☐ ▸ user		
✓ ▸ views		
✓ ▾ bios		
✓ retrieve	Retrieves a view.	

Figure 4. Example Services Server

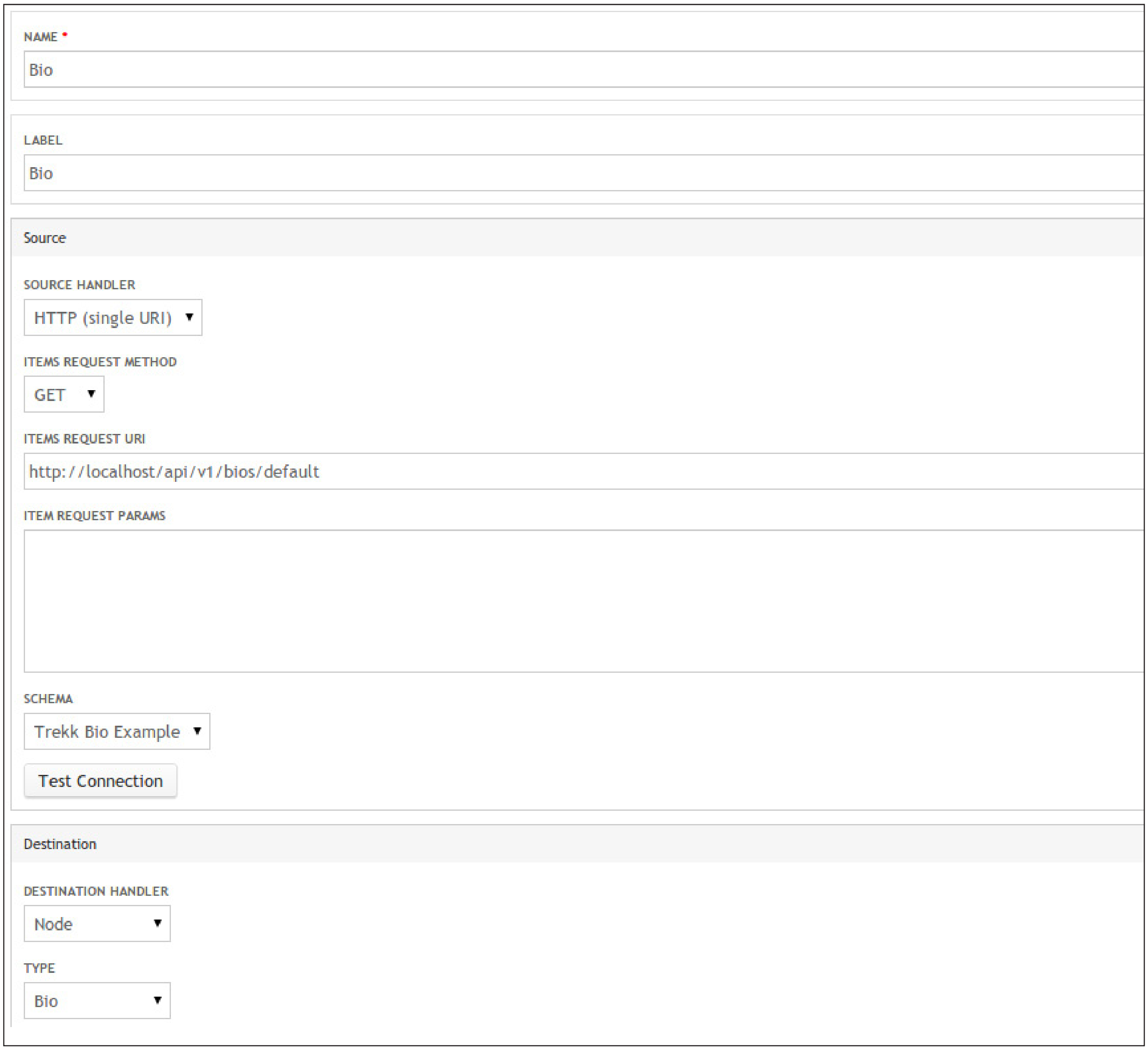
Listing 1. Trekk Client Source Snippet

```
/**
 * Implements hook_trekk_client_source_schema()
 */
function bio_example_trekk_client_source_schema() {
  $schema = array();

  $schema['bio_example'] = array(
      'name' => 'bio_example',
      'label' => t('Trekk Bio Example'),
      'fields' => array(
        'nid' => array(
          'type' => 'int',
          'length' => 11,
          'unsigned' => TRUE,
          'not null' => TRUE,
          ),
        'node_title' => array(
          'type' => 'varchar',
          'length' => 255
          ),
        'body' => array('type' => 'longtext'),
        'position' => array(
          'type' => 'varchar',
          'length' => 255
          ),
        ),
      'primary key' => array('nid'),
      );

  return $schema;
}
```

"http://localhost/api/bios/default" where "api" is the Services path, "bios" is the View name, and "default" is the View display. You also can add query string parameters like "output=html" to show HTML. Note, other examples and details are contained within the Trekk distribution. The Trekk Server is now set up.

Covering the creation of a Trekk Client in detail is beyond the scope of this article; instead, I review the basic process. First, create a second instance of Trekk, this time selecting the "Trekk Client" option. Trekk clients use Migrate and need custom code to parse and import the data that's retrieved from the server. So, each Trekk Client has custom code tailored to the shared content. This code is composed of three primary parts: a Source, a Destination and a Mapping. (Note: there are examples within the Trekk distribution.)

You'll need to navigate to admin/config/services/trekk_client and add a new Trekk Client configuration item. You can define the Trekk


NAME *

Bio

LABEL

Bio

Source

SOURCE HANDLER

HTTP (single URI)

ITEMS REQUEST METHOD

GET

ITEMS REQUEST URI

http://localhost/api/v1/bios/default

ITEM REQUEST PARAMS

SCHEMA

Trekk Bio Example

Test Connection

Destination

DESTINATION HANDLER

Node

TYPE

Bio

Figure 5. Example Bio Configuration

Server URL as well as the HTTP method, but the Source schema implements hook_trekk_client_source_schema(). The schema is essentially a database-friendly version of the structure the Trekk Server endpoint outputs. For example, your bio schema would have "nid", "node_title", "body" and "position" as the "fields", and each "field" would have the data type (int, varchar and so on) and length where appropriate—see Listing 1 for an illustration. The only hitch is that the Trekk Server converts a View into JSON, so it uses the View field labels instead of the machine name as you have in other areas—hence, "position" and not "field_position". Once you have configured the Source, you can "Test Connection" and verify that the Trekk Client can access the Trekk Server, although this does not validate the schema.

Trekk extends Migrate's node and term Destinations to make them accessible to the GUI. However, many contrib modules provide Migrate Destinations, and Migrate maintains several as well in the migrate_extras module. For example, the Flatfish module media Migrations use the Media Destination to create the physical files and populate the file_managed table properly. In many cases, you will be able to use the node Destination. For complex Migrations, you may need to extend Destinations or write your own.

Finally, you will need to add a Mapping. Although Mappings can be very simple, they are more than just the connection between the Source and the Destination. Mappings also provide a key juncture to insert custom processing—you can register custom callbacks. For example, although we use the standard node Destination in the Flatfish module Content Type Migrations, a callback executes more than 100 lines of custom code. This is the code that cleans the HTML, handles the non-Media file creation (PDFs, docs and so on) and replaces the Flatfish tokens with Media tokens. One last note on the Mappings—Trekk handles Mappings a little differently from normal Migrations and expects Mappings to come in as configuration rather than as part of a Migration class constructor. For a full understanding, I recommend reviewing TrekkClientMapping.inc in detail.

You now should have content on the Trekk Client. From here, just write a simple drush script to execute Migrate updates and set up a cron job. The Trekk Client content will be up to date and largely maintenance-free.

As you can see from this overview and example, Trekk provides a great starting point for universities looking to leverage Drupal in a big way.■

Tim Loudon is the VP of Engineering at Drupal Connect, a full-service Drupal development shop. He has been working with Drupal since 2008 and has had the opportunity to work with some fantastic clients on some amazing sites. Recently, Tim has slung code for the Stanford School of Engineering and the Stevens Institute of Technology.

Resources

Trekk Home Page:
http://drupalconnect.com/trekk

The Flatfish Repo:
https://github.com/drupalstaffing/flatfish

The Flatfish Module:
http://drupal.org/project/flatfish

Trekk IRC Channel: #trekk

Creating and Theming a Custom Content Type with Drupal 7

Make the most of your content strategy by building and theming custom content types using Drupal 7. In this article, I show how to design an Event content type, leveraging some features with this content management framework.

DANNY ENGLANDER

One of the great new things about Drupal 7 is that it's now easier to customize your site content. In Drupal 6, you typically had to use the CCK (Content Construction Kit) module for fine-grained control in customizing content, but that has been folded into core for Drupal 7. Drupal 7 is now a true content management framework (CMF).

Drupal 7: It's All about Fields

When you customize content in Drupal 7, it involves creating or modifying what's often referred to as a *content type*. Drupal 7 comes with two defaults: Page and Article. When building a site, you often need additional content types. These might include events, press releases, FAQ, staff, photo gallery and more. What makes these unique? It's all about the *Fields* that you add to your content type.

A Field is an attribute for the content type, and these attributes are types of information associated with your content. If you have an Event content type, you might need to add date, location and link attributes. Each of those can be realized through the use of Fields.

Theming and Nodes

A Drupal 7 Node is simply an individual page with content that's rendered using a specific content type that contains

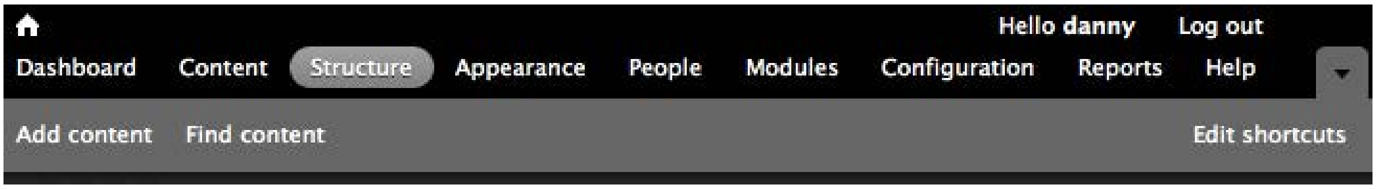
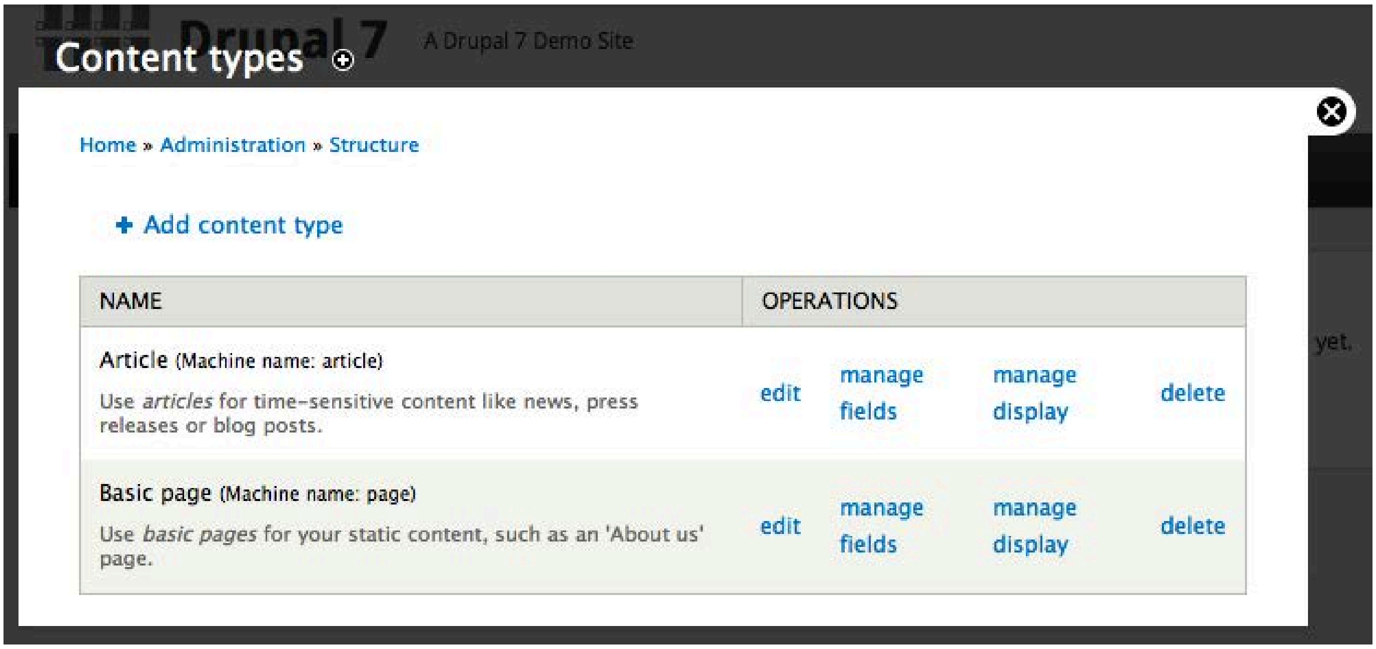


Figure 1. Drupal 7 Administrator Toolbar



NAME	OPERATIONS			
Article (Machine name: article) Use *articles* for time-sensitive content like news, press releases or blog posts.	edit	manage fields	manage display	delete
Basic page (Machine name: page) Use *basic pages* for your static content, such as an 'About us' page.	edit	manage fields	manage display	delete

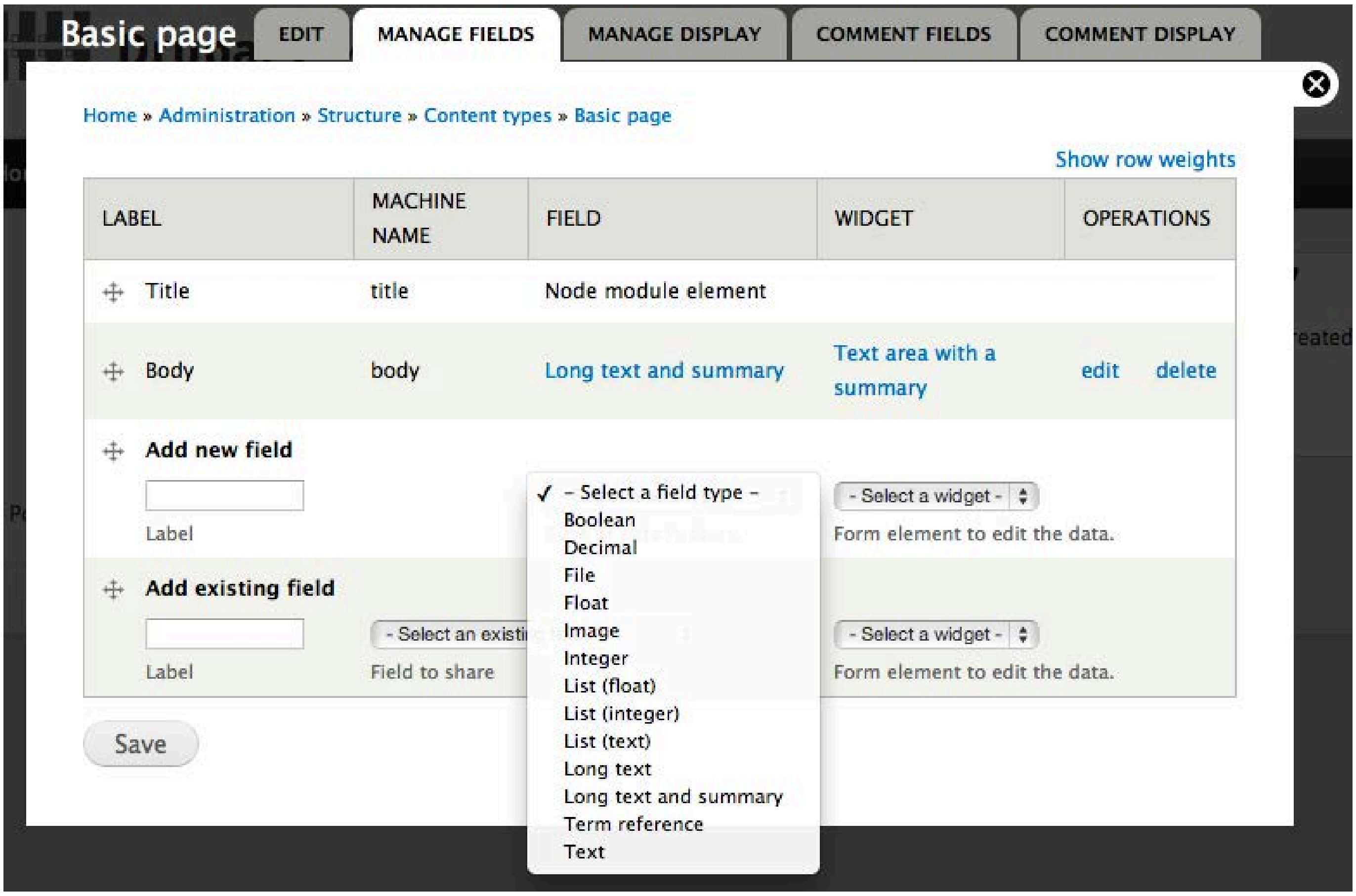
Figure 2. The content type landing page is where you can edit your content types and manage their fields and display.

all of your fields. Fields are extremely flexible, and there are many possibilities for display and theming them, depending on what you want to do. Theming, in regard to Drupal, is the presentational layer of the code and content, and there are various ways to achieve theming from some simple changes in the Drupal Admin UI to more complex custom node templates—for example, node--custom.tpl.php. In this tutorial, I use the Display Suite module for theming our node. It's a visual interface that allows for custom layouts without really needing to know too much code. There also are other ways to theme a node, including custom node templates and the Panels module.

Drupal 7 comes with basic default fields, such as Text, Long text, File, Image and List. There also are many additional Field types that can be added via contributed (contrib) modules.

Manage and Edit

To gain insight into existing content types and Fields on your site, navigate to /admin/structure/types, or use the admin toolbar: Structure→Content types.



Basic page | EDIT | MANAGE FIELDS | MANAGE DISPLAY | COMMENT FIELDS | COMMENT DISPLAY

Home » Administration » Structure » Content types » Basic page

Show row weights

LABEL	MACHINE NAME	FIELD	WIDGET	OPERATIONS
Title	title	Node module element		
Body	body	Long text and summary	Text area with a summary	edit delete
Add new field Label		✓ - Select a field type -	- Select a widget - Form element to edit the data.	
Add existing field Label	- Select an existi Field to share		- Select a widget - Form element to edit the data.	

Figure 3. You can manage existing fields for an individual content type and add new ones with a select menu.

Figure 1 shows the administrator toolbar at the top of the Admin Drupal UI in the browser. You'll use this a lot to navigate the admin area for various tasks. This assumes you have admin access to a Drupal site or at least have been given the proper permissions by an admin to see and use the toolbar.

On the content type landing page, as shown in Figure 2, you can see existing types, and you would add new ones here.

You also can edit a content type, add and edit Fields, and manage its display from this page. Figure 3 shows existing Fields and a select menu to add new ones for a content type.

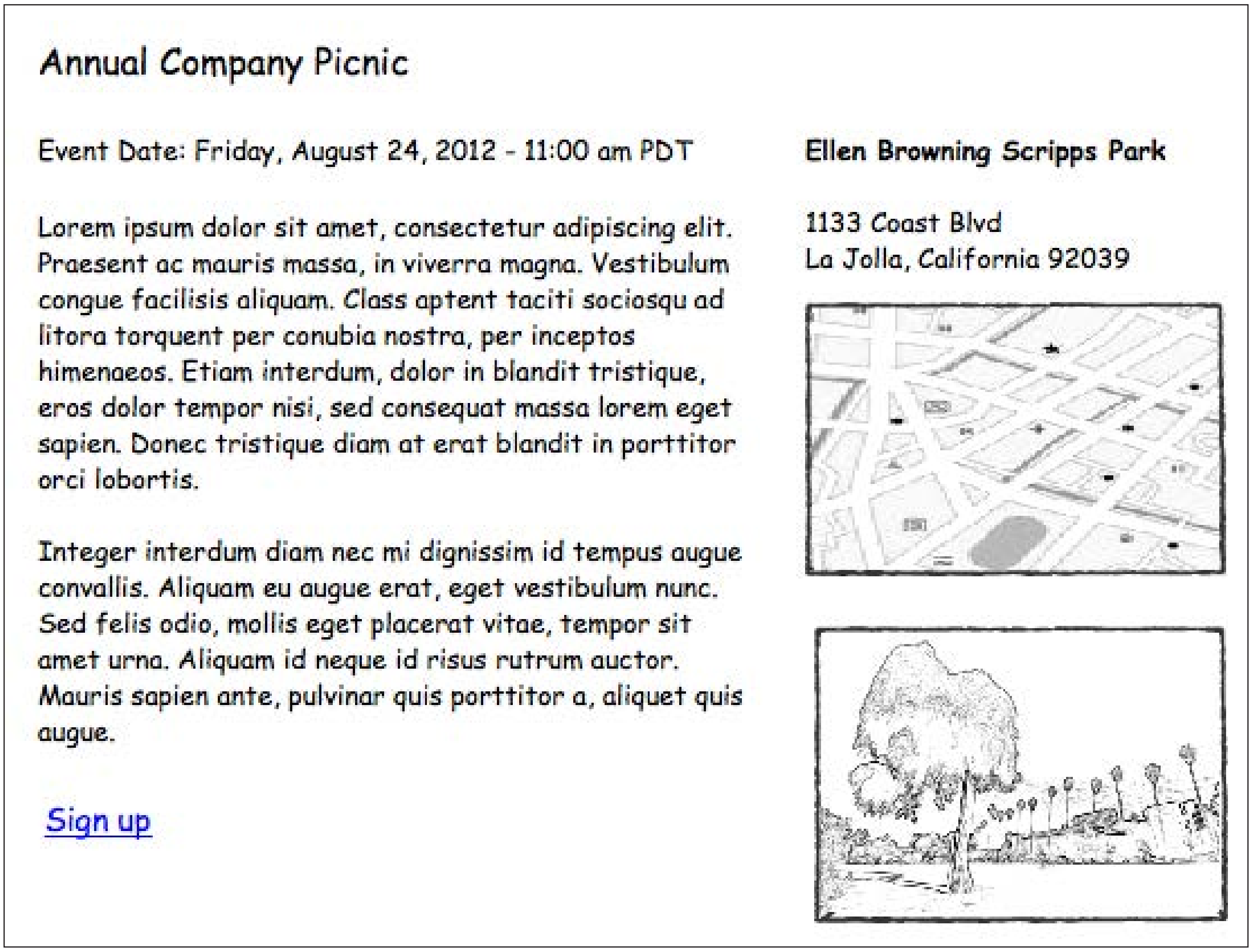
Tools Needed

Now that you have a basic overview of how content types work, let me show you how to build a new one and theme it. For this example, let's create and theme an Event content type. You will need a few additional contrib modules available from drupal.org. Ideally, to follow along with the tutorial, you'd want a fresh install of a Drupal 7 site, and download and set the default theme to the Professional Theme.

If you are not familiar with how to install Drupal, it's actually not that hard, and you can install it in a LAMP environment. If you already have a local development machine with LAMP, it's a matter of creating a new MySQL database and following a point-and-click GUI install for Drupal. There are links at the end of this article in the Resources section for some basic Drupal install information. Note that Drupal has an entire command-line utility of its own called drush where you can install core and download contrib modules, but for the sake of this tutorial, I'm using the Drupal Admin UI. If you really decide to dive in head first to Drupal, drush can be a huge time saver.

Mock It Up

When creating a new content type, I like to do a quick mock-up to get a better picture of how I want the finished product to look. Figure 4 shows what I came up with for my Event.



Annual Company Picnic

Event Date: Friday, August 24, 2012 - 11:00 am PDT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent ac mauris massa, in viverra magna. Vestibulum congue facilisis aliquam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Etiam interdum, dolor in blandit tristique, eros dolor tempor nisi, sed consequat massa lorem eget sapien. Donec tristique diam at erat blandit in porttitor orci lobortis.

Integer interdum diam nec mi dignissim id tempus augue convallis. Aliquam eu augue erat, eget vestibulum nunc. Sed felis odio, mollis eget placerat vitae, tempor sit amet urna. Aliquam id neque id risus rutrum auctor. Mauris sapien ante, pulvinar quis porttitor a, aliquet quis augue.

Sign up

Ellen Browning Scripps Park

1133 Coast Blvd
La Jolla, California 92039

Figure 4. Mock-up of how an Event page will look.

Essential Modules

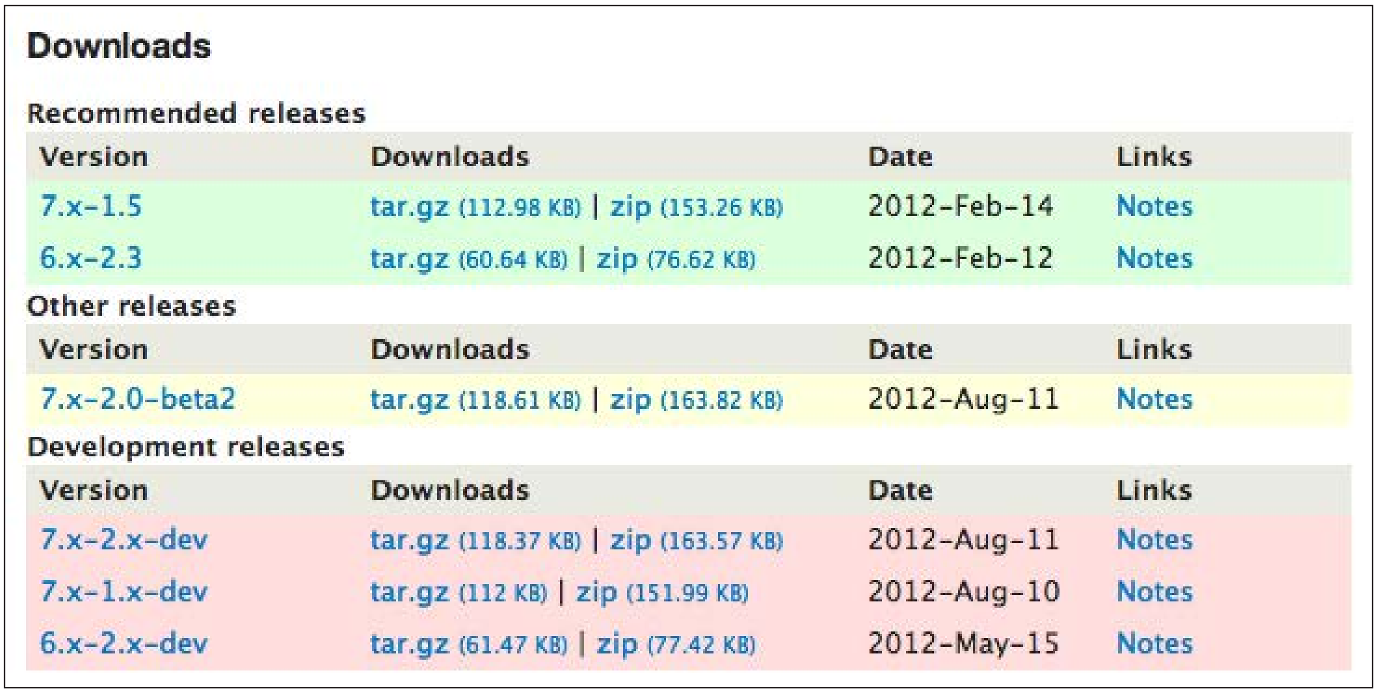
The mock-up helps determine which modules I'll use. You can find links for these in the Resources section. Contributed modules and Theme (contrib) required the following:

- Date
- Get Locations
- Geocoder
- GeoPHP
- Chaos tool suite (ctools)
- Link
- Libraries
- Display Suite
- Professional Theme

Core Fields—included with Drupal 7 (no need to download):

- Image
- Overview (Long Text/Textarea)

Now that you know what modules and Theme you are going to use for building your Event content type, download the latest stable *recommended* release from drupal.org (highlighted in green on its project page). The recommended release



Downloads

Recommended releases

Version	Downloads	Date	Links
7.x-1.5	tar.gz (112.98 KB) | zip (153.26 KB)	2012-Feb-14	Notes
6.x-2.3	tar.gz (60.64 KB) | zip (76.62 KB)	2012-Feb-12	Notes

Other releases

Version	Downloads	Date	Links
7.x-2.0-beta2	tar.gz (118.61 KB) | zip (163.82 KB)	2012-Aug-11	Notes

Development releases

Version	Downloads	Date	Links
7.x-2.x-dev	tar.gz (118.37 KB) | zip (163.57 KB)	2012-Aug-11	Notes
7.x-1.x-dev	tar.gz (112 KB) | zip (151.99 KB)	2012-Aug-10	Notes
6.x-2.x-dev	tar.gz (61.47 KB) | zip (77.42 KB)	2012-May-15	Notes

Figure 5. The latest recommended release of a module is shown highlighted in green (7.x-1.5 in this case). Dev releases are red, and a new 2.x beta branch is yellow.

should be sufficient, but there may be times when you run into a bug and need to use the latest *dev* release. Figure 5 shows an example of release information for a contrib module on drupal.org.

Download the modules, untar them, and put the resulting folders in the /sites/all/modules folder from Drupal root. Although you will see a Modules folder at root, never add third-party modules to this one; it's a core folder that gets updated or changed only when a core Drupal release comes out. Likewise, download the Professional Theme, untar it, and put it in /sites/all/themes.

You also need to download some map markers from the Get Locations project page. The getlocations-markers.tar.gz file should be sufficient. Untar this file, and put it in sites/all/libraries/getlocations/markers. Most likely you will have to create the Libraries folder on a new Drupal install, but if it exists already, simply make a new folder within that called getlocations, and then put the newly untarred Markers folder within that.

Activate Contrib Modules and Theme

Go to the Drupal Modules Admin page located at /admin/modules, or from the admin toolbar, click on Modules. Check the check boxes for Get Locations, Getlocations Fields, geoPHP, Geocoder, Libraries, Chaos tools, Link, Date, Date API, Date popup and Display Suite, and then click Save configuration.

Using the admin toolbar, click on Appearance, and activate the Professional Theme listed under Disabled Themes. Use the Enable and set default setting that will enable it and bring it up to the top under Enabled Themes.

Get a Google API v3 Key

You'll need to add a Google Maps JavaScript API v3 Key to display a map for an event location. Go to the Google API console and log in with your Google account (or create a Google account if you haven't already). See Resources for links. Once logged in, generate a new key, or use an existing one from the API Access tab. Back in Drupal, use the admin toolbar to go to Configuration, and then under Web Services, click on Get Locations. Expand Google API Key, enter your key, and save the configuration.

Set Your Locale and Create a Custom Date and Time Format

Now is a good time to set your site's Locale as well. From the toolbar, go to Configuration→Regional and Language→Regional settings or admin/config/regional/settings. Chances are your default time zone already is set, but it's good to confirm or adjust it. Accept all other defaults here and save. Also, from Regional and

Language, click on Date and time, located at admin/config/regional/date-time, and then the Formats tab. Let's create a custom date format. It's pretty straightforward, and you'll use standard code from the PHP Date Manual (see Resources). Once on the Date and Time formats page, click Add Format. In the Format string text box, copy and paste (or type) in this string:

```
l F j, Y-  g:i a T
```

You will see that it immediately renders as a nicely formatted date like this:

```
Sunday August 12, 2012- 11:00 am PDT
```

Let's give this new format a name. Go back to the main Date and Time page you were just on, and click Add date type. Let's call it Full date with time zone for Date type. For Date Format, choose from the select menu, and most likely the new format you just created will be the last item in the menu. Choose that, and save (Add date type). The new Date Type is saved (you'll refer to it later).

Create an Image Style

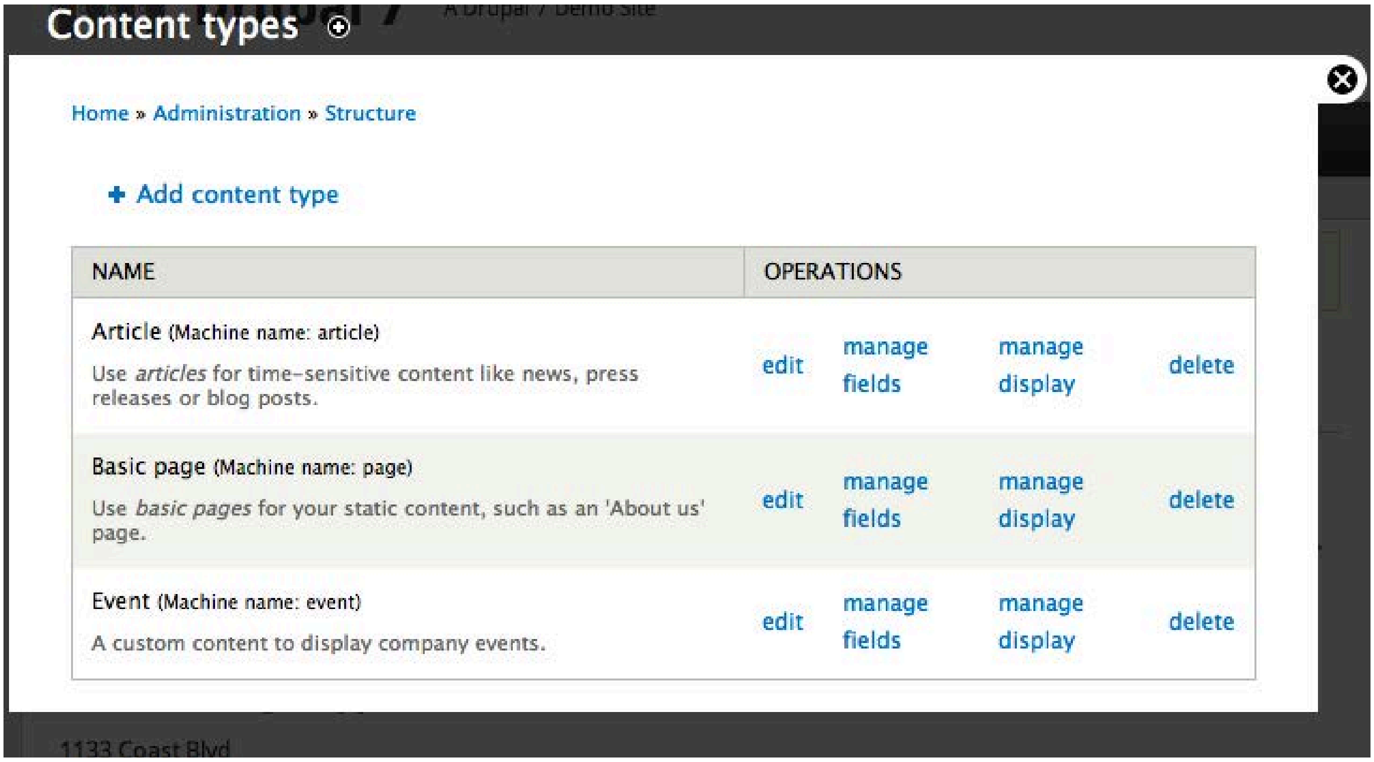
For the event image, you'll want a custom style. Using the admin toolbar, click on Configuration and then Media→Image Styles or go to /admin/config/media/image-styles. Click Add Style, type in `event_photo`, and save. Under Effects, choose Scale in the Select New Effect select list. For width, choose 375, and save (Add effect). The new effect is saved, and you will refer to it later.

Build It

Now, go to Structure→Content types from the admin toolbar, and click Add content type. You now get a new page with options. You can pretty much accept the defaults here, but let's customize a little:

- Name: Event.
- Description: a custom content to display company events.
- Publishing Options: uncheck Promoted to front page.
- Display Settings: uncheck Display author and date information.
- Comment Settings: set to Hidden under Default comments setting for new content.
- Menu settings: uncheck Main Menu.

Save the content type. Back on the content types landing page, you now will see Event (Machine name: event), in addition to Basic page and Article, as shown in Figure 6.



Content types

Home » Administration » Structure

+ Add content type

NAME	OPERATIONS			
Article (Machine name: article) Use *articles* for time-sensitive content like news, press releases or blog posts.	edit	manage fields	manage display	delete
Basic page (Machine name: page) Use *basic pages* for your static content, such as an 'About us' page.	edit	manage fields	manage display	delete
Event (Machine name: event) A custom content to display company events.	edit	manage fields	manage display	delete

Figure 6. The Newly Added Event Content Type

Add Fields

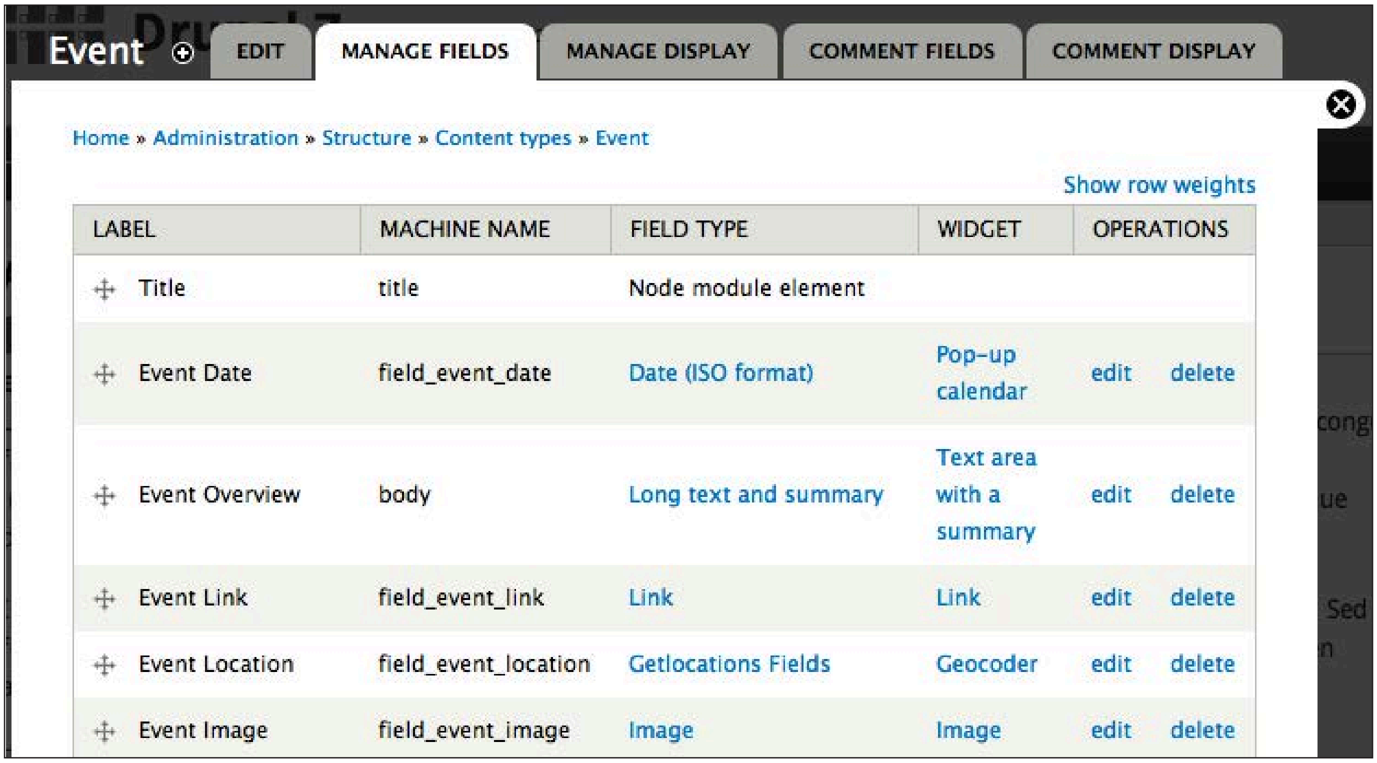
Now you're ready to add/edit your custom event fields. Click manage fields for Event. You'll see that the Body Field already exists. Click to edit on Body, and simply change the Label to Event Overview, and save the settings.

Add a Date Field:

1. On Event→Manage Fields under Add new field, type in `Event Date`.
2. Type — Date (ISO format).
3. Widget — Pop-up calendar.
4. Click Save.
5. Now on Field Settings, accept the defaults and save again.
6. On the next screen, check Required Field.
7. Click on More Settings and Values.
8. Now for Date Entry Options, choose a format from the select list. Ideally, choose something that shows the day, month, year and time with the desired time format (whether it's 24-hour time or am/pm)—for example, Aug 10 2012 - 3:30:13pm.

Add a Location Field:

1. On Event→Manage Fields under Add new field, type in Event Location.
2. Type — Geolocations Fields.
3. Widget — Geocoder.
4. Save, and on the next screen for Search options, choose No Search box.
5. Save, and on the next screen, check Required field.



Event | EDIT | MANAGE FIELDS | MANAGE DISPLAY | COMMENT FIELDS | COMMENT DISPLAY

Home » Administration » Structure » Content types » Event

Show row weights

LABEL	MACHINE NAME	FIELD TYPE	WIDGET	OPERATIONS	
Title	title	Node module element			
Event Date	field_event_date	Date (ISO format)	Pop-up calendar	edit	delete
Event Overview	body	Long text and summary	Text area with a summary	edit	delete
Event Link	field_event_link	Link	Link	edit	delete
Event Location	field_event_location	Getlocations Fields	Geocoder	edit	delete
Event Image	field_event_image	Image	Image	edit	delete

Figure 7. The finished Fields page for Event. You can re-order the fields by dragging the little + icons under the label, but you must save the page after doing so.

6. Accept all other defaults here, and save again.

Add a Link Field:

1. On Event→Manage Fields under Add new field, type in `Event Link`.
2. Type — Link.
3. Widget — Link.
4. Save, and on the next screen, simply save again.
5. On the next screen, make required and choose Required Title, and save.

Add an Image Field:

1. On Event→Manage Fields under Add new field, type in `Event Image`.
2. Type — Image.
3. Widget — Image.
4. Save, accept defaults, and save again.
5. On the next screen, check Required field.
6. Check the Alt and Title fields, and save.

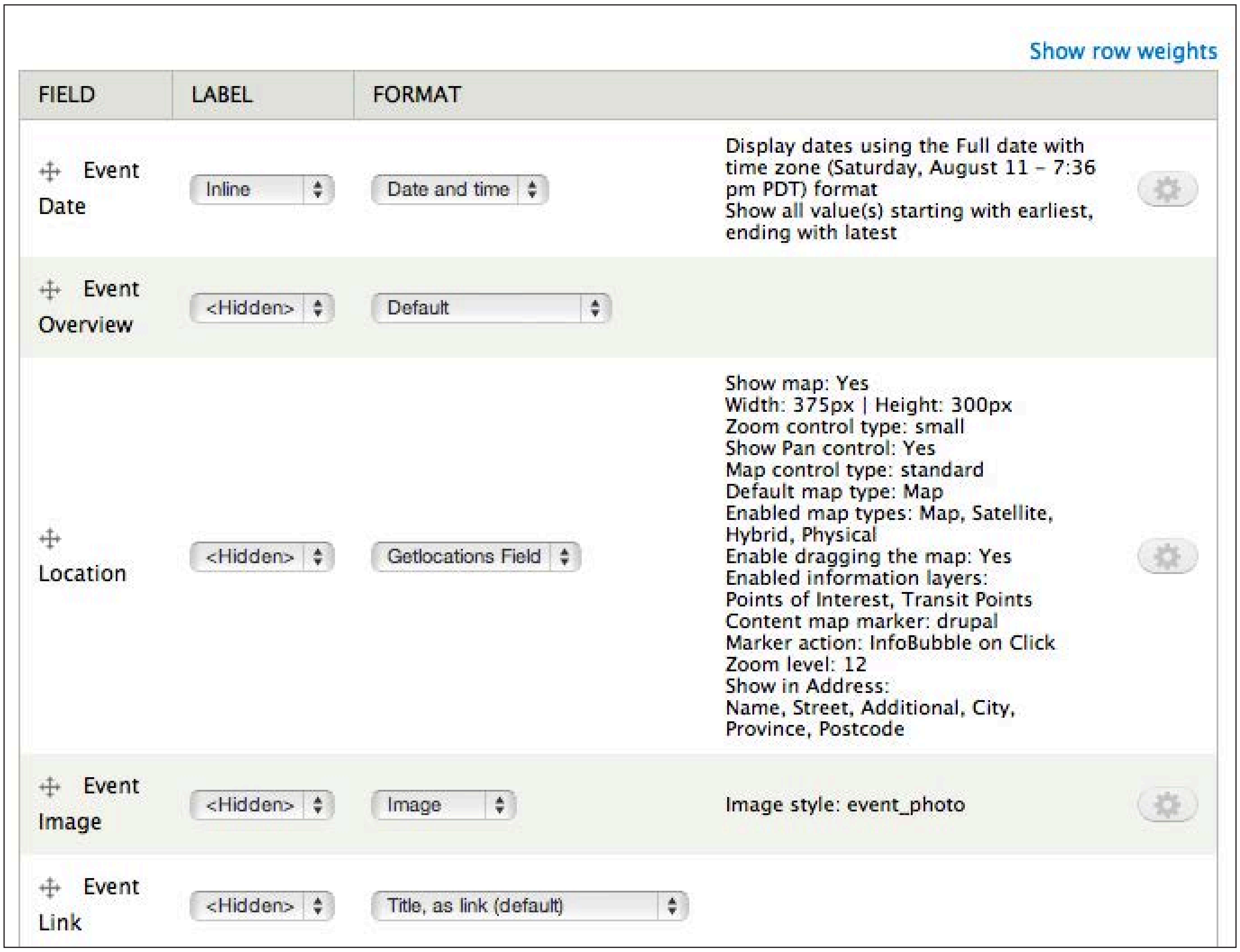
Figure 7 shows the finished Manage Fields page. Note that you can drag the fields around to re-order them with the little + icons, but when you do, you need to save the page again.

As you might have noticed when creating all these fields, there are dozens of other options for customization, and for the most part, you accepted many of the defaults. However, as you get more into Drupal site building, you'll want to explore these options and experiment with them.

Note that you can drag the fields around to re-order them with the little + icons, but when you do, you need to save the page again.

Manage Your Display

Now you just need to do a little theming using the Manage Display tab for your Event content type. This will be a two-part process. The first is to get some reasonable defaults for your display. Using the admin toolbar, head back to Structure→Content types→Event→manage display. Using the "select lists and drag" icons, make your layout to look like Figure 8. Note the Gear icons at the



Show row weights

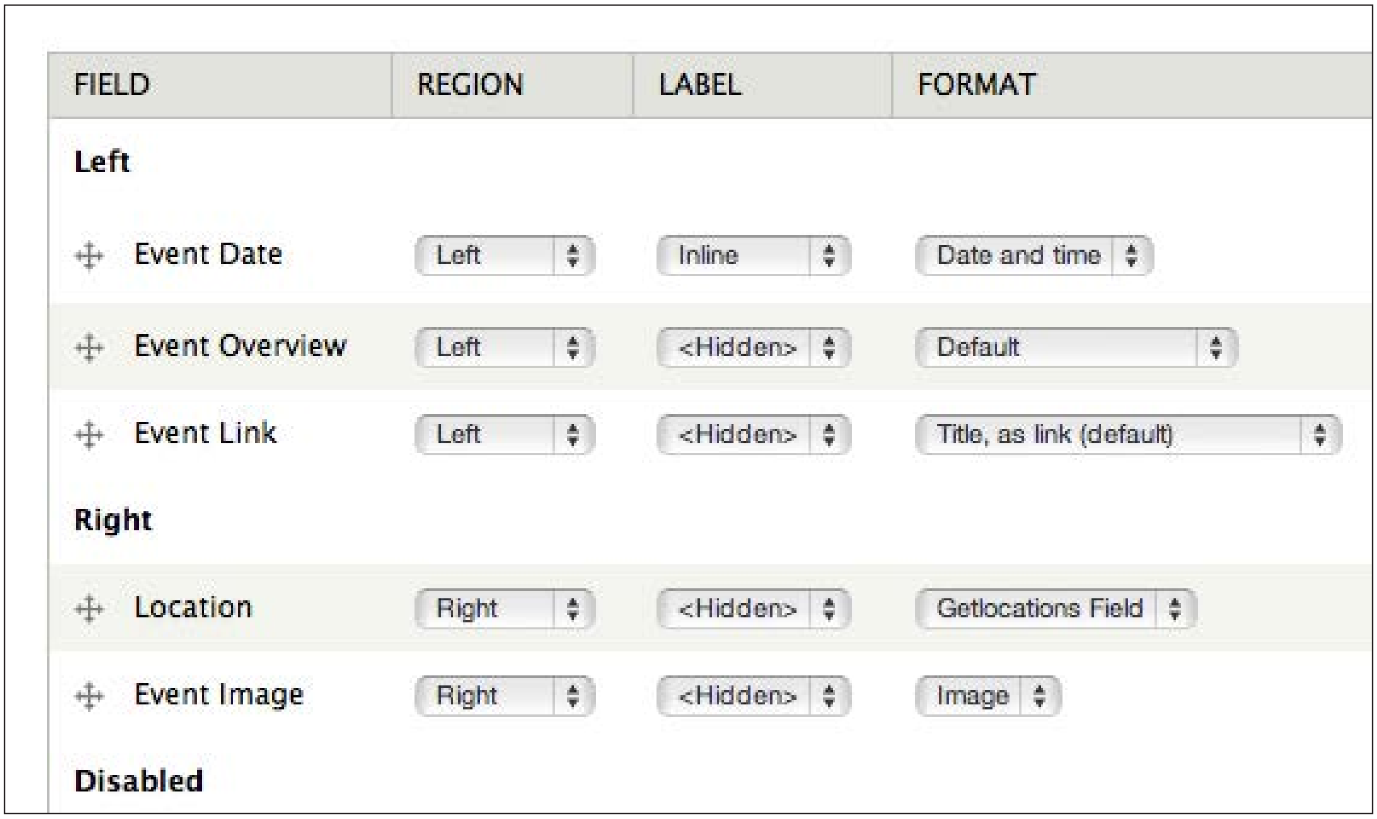
FIELD	LABEL	FORMAT		
Event Date	Inline	Date and time	Display dates using the Full date with time zone (Saturday, August 11 - 7:36 pm PDT) format Show all value(s) starting with earliest, ending with latest	
Event Overview	<Hidden>	Default		
Location	<Hidden>	Getlocations Field	Show map: Yes Width: 375px | Height: 300px Zoom control type: small Show Pan control: Yes Map control type: standard Default map type: Map Enabled map types: Map, Satellite, Hybrid, Physical Enable dragging the map: Yes Enabled information layers: Points of Interest, Transit Points Content map marker: drupal Marker action: InfoBubble on Click Zoom level: 12 Show in Address: Name, Street, Additional, City, Province, Postcode	
Event Image	<Hidden>	Image	Image style: event_photo	
Event Link	<Hidden>	Title, as link (default)		

Figure 8. The Configured Layout after Adjusting Labels and Settings

Note the Gear icons at the far right; you can click on those to fine-tune your settings, but always remember to choose Update, and then save the page after any changes.

far right; you can click on those to fine-tune your settings, but always remember to choose Update, and then save the page after any changes. You'll notice with the Date and Event photo, you now have the respective format and style you created previously.

Let's add in a Display Suite layout; this is where all the magic happens. At the bottom of Manage Display, find the tab called Layout for event in default. Click, and using the Select a layout list, choose Two Column, and save the page. You'll notice you have an updated UI showing Right, Left and Disabled. All Fields default to Disabled, so position the Fields where you want them in your new two-column node layout. You either can drag your Fields



FIELD	REGION	LABEL	FORMAT
Left			
Event Date	Left	Inline	Date and time
Event Overview	Left	<Hidden>	Default
Event Link	Left	<Hidden>	Title, as link (default)
Right			
Location	Right	<Hidden>	Getlocations Field
Event Image	Right	<Hidden>	Image
Disabled			

Figure 9. The Finished Display Suite Layout

up to the right or left area or use the Region select list to choose. I find the latter to be easier. Figure 9 shows the layout after repositioning Fields and saving the page.

Add CSS to the Theme

By default, the two-column Display Suite layout is split 50%, but let's add some CSS to our theme to mirror the wireframe. Open styles.css, located at sites/all/themes/professional_theme/, and add this code after the final closing brace at the end of the file. You'll notice this CSS file has media queries in it, and for the sake of this demo, let's add our styles in after any media query, but you also could theme this per media query:

```
.group-right {
width: 36%;
padding-left: 2%;
float: right;
}

.group-left {
width: 62%;
}

.field-name-field-event-image {
margin-top: 20px;
}
.getlocations_map_canvas {
max-width: 100%;
}
```

From the admin toolbar, click on Content, and then choose Add content, and choose Event. You now will see a blank node create form. Fill in a Title, choose your Date, and fill in some Overview Text. Next, choose an address for your event, and once it's filled in, click the Geocode this address button, and your little map below will re-center itself on your address. Finally, upload an image, choose a link/title, and save your node. Figure 10 shows the final themed layout.

FIBERCLOUD®
Growth at Any Speed

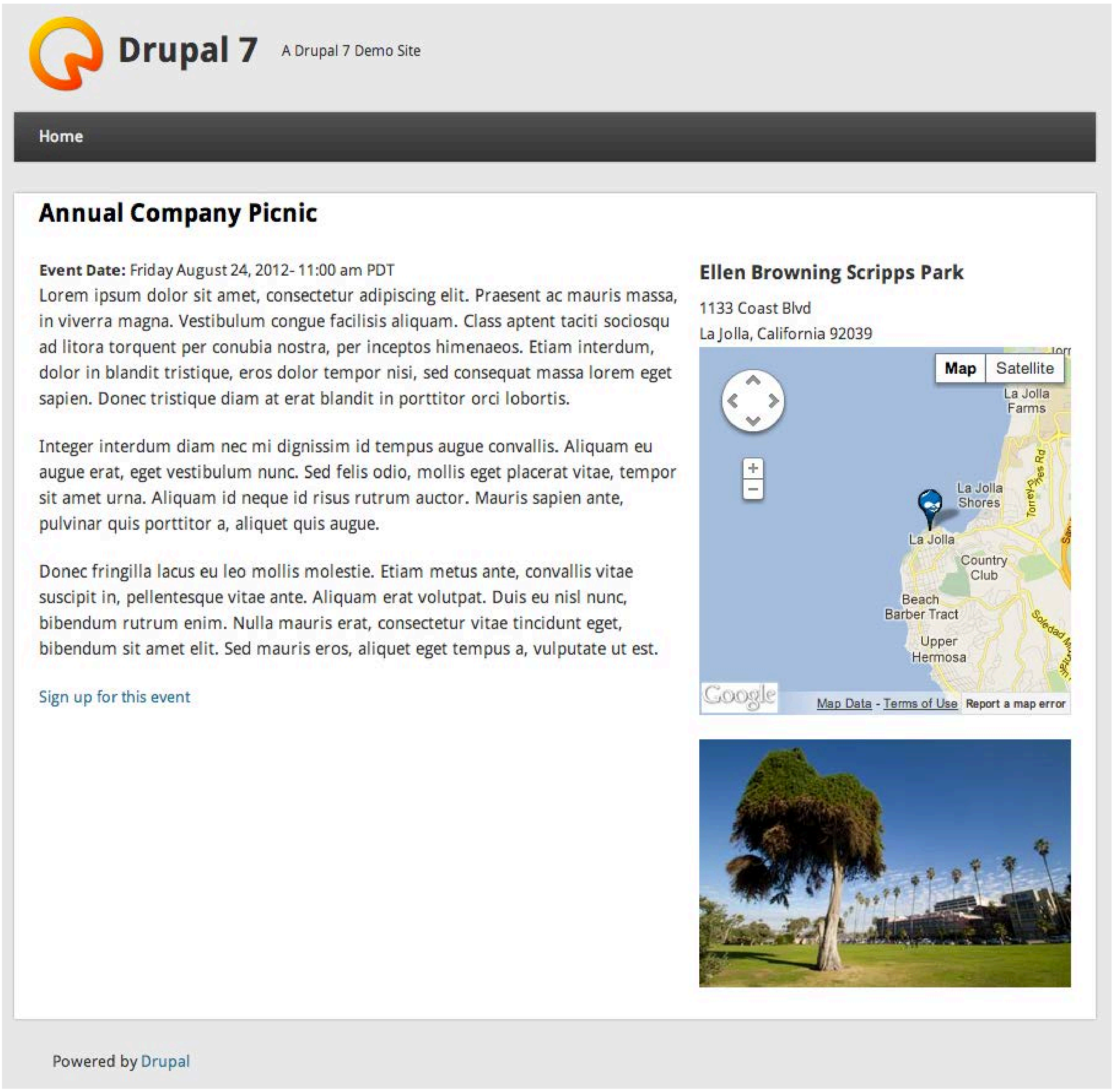
CLOUD SERVERS FOR DRUPAL DEVELOPERS

DRUPAL EXPERTS

You have options. Choose the Linux hosting solution that fits your needs. Secure, reliable and flexible infrastructure packaged with you in mind.

Secure, Reliable, Supported, Flexible, Easy

FiberCloud
1.877.926.3282
www.fibercloud.com/drupal-servers



Drupal 7 A Drupal 7 Demo Site

Home

Annual Company Picnic

Event Date: Friday August 24, 2012- 11:00 am PDT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent ac mauris massa, in viverra magna. Vestibulum congue facilisis aliquam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Etiam interdum, dolor in blandit tristique, eros dolor tempor nisi, sed consequat massa lorem eget sapien. Donec tristique diam at erat blandit in porttitor orci lobortis.

Integer interdum diam nec mi dignissim id tempus augue convallis. Aliquam eu augue erat, eget vestibulum nunc. Sed felis odio, mollis eget placerat vitae, tempor sit amet urna. Aliquam id neque id risus rutrum auctor. Mauris sapien ante, pulvinar quis porttitor a, aliquet quis augue.

Donec fringilla lacus eu leo mollis molestie. Etiam metus ante, convallis vitae suscipit in, pellentesque vitae ante. Aliquam erat volutpat. Duis eu nisl nunc, bibendum rutrum enim. Nulla mauris erat, consectetur vitae tincidunt eget, bibendum sit amet elit. Sed mauris eros, aliquet eget tempus a, vulputate ut est.

Sign up for this event

Ellen Browning Scripps Park

1133 Coast Blvd

La Jolla, California 92039

Powered by Drupal

Figure 10. The Final Themed Layout for the Event Content Type

There's so much more you can do with custom content types and theming in Drupal 7; this is just the tip of the iceberg. To enhance the content editor experience, a nice addition would be to add a WYSIWYG editor to the Overview Text area. If you want to get more hands-on with code, you could dispense with Display Suite and add a node--event.tpl.php file to your theme folder and theme the Fields individually.■

Resources

Professional Theme (front-end theme pictured in the demo): **http://drupal.org/project/professional_theme**

Date: **http://drupal.org/project/date**

Get Locations: **http://drupal.org/project/getlocations**

Link: **http://drupal.org/project/link**

Geocoder: **http://drupal.org/project/geocoder**

geoPHP: **http://drupal.org/project/geophp**

Display Suite: **http://drupal.org/project/ds**

Libraries API: **http://drupal.org/project/libraries**

Get Locations Documentation:
http://drupalcode.org/project/getlocations.git/blob/refs/heads/7.x-1.x:/README.txt

Installing Drupal 7: **http://drupal.org/documentation/install**

Installation Tutorial Video: **http://learnbythedrop.com/drop/173**

Sign in/Create a Free Google Account: **https://code.google.com/apis/console**

Google Maps API Key (v.3.0):
https://developers.google.com/maps/documentation/javascript/tutorial#api_key

Drupal Is Not a CMS: **http://www.palantir.net/blog/drupal-not-cms**

Danny Englander is a Drupal Developer specializing in theming, site building, UX, UI, responsive design and JQuery. He runs his own Drupal shop and freelances for various clients around the United States. He has a real passion for Drupal and likes getting involved with the Drupal community and on drupal.org to help out in the issue queue and forums. He recently contributed his first module to drupal.org, and also started designing and developing a contrib theme. Danny lives in San Diego, California, with his wife Elise, and together they love traveling and exploring. He is an avid photographer and recently designed and developed a new photography Web site, http://highrockphoto.com, built with Drupal, of course. For more information, see http://highrockmedia.com or http://twitter.com/highrockmedia.

Tips for Writing Interoperable Drupal Distributions

Want a crash course on how to write distributions for the Drupal platform? Get tips and expert advice from a longtime Drupal contributor and distribution developer. This article is packed with examples of problems and solutions for writing plugins that integrate cleanly with other distributions and make the best use of the Drupal toolkit.

NEDJO ROGERS

First off, what is a distribution and why would you write one? Even more than many other content management system (CMS) frameworks, Drupal is up for pretty much any task you set it. You can think of Drupal like a potter's workshop. Want a vase or plate or mug or Halloween mask? You can find different kinds of clay on the shelf and the potter's wheel is right over there—go for it!

No problem—as long as you're a skilled potter. But, if it's your first time spinning clay, your plate's likely to come out wonky or crack when it hits the kiln.

What you want is a mold—something that'll give your plate just the right thickness and structure and shape. From there, sure, you can add your own details and dab on whatever glaze you like.

So if Drupal is like a potter's workshop, a distribution is the plaster mold—something you can use quickly to make an expertly designed product that fits a particular purpose.

There's a growing set of Drupal distributions available. Need to build a Web site for a conference, newspaper, government agency, on-line store or nonprofit organization? Rather than starting from scratch, you'll get a huge leg up if you start with the Conference Organizing Distribution (COD), OpenPublic, OpenPublish, Commerce Kickstart or Open Outreach.

Why Write a Distribution?

Maybe the first question should be, "what are some top reasons *not* to write a distribution?" Fame and fortune, for one—writing and maintaining a distro is hard work. It eventually may create some income in the form of contracts to extend existing functionality, but in the short term, you've got a ton of designing and coding to get through. As for fame, well, despite their merits, most distributions are notably modest in terms of both community profile and the number of actual installs.

Another good reason not to write a distribution is there already may be one (or more) in existence that fits the bill. True, those existing distributions may not do everything you want them to do exactly the way you would have done it. But, rather than doing it your own way, try pitching in and improving the existing projects. That's what open source is all about.

With the "why nots" out of the way, here are some key reasons to consider writing a distribution:

- You're working with or for a large institution or network that will need many sites built along a common model.
- You've got a sizable contract to build a Web site that fits a need not already covered by an existing distribution; you expect to be doing similar sites in the future, and you've got room in the project to generalize.
- You're tired of doing the same site-building work over and over and want to focus on refinements.
- You have a burning interest in and commitment to a particular sector or need and want to dig in over the long term.
- You're committed to lowering the financial and technical barriers to technology access.
- You're up for sharing your work, even though doing so will feel like pinning a badge to your chest labeled "complaints department".

- You're an open-source evangelist and deeply believe that everyone is best served when technology is freely available to all.

Key Concepts in Distribution Authoring

Before I go into details, here's a quick terminology overview:

- A *module* is a plugin that extends Drupal's functionality. The core Drupal product, *Drupal core*, ships with a number of modules, and community contributors have written thousands more.

- An *install profile* is a special kind of Drupal module that contains code to be used only when the site is first installed. Each distribution needs its own install profile.

- A *feature module* or *feature* is a Drupal module produced with the Features module and containing exported configuration. A distribution typically is made up of multiple feature modules, each providing a distinct set of functionality.

- The features approach is built on *exportables*—APIs that allow configuration from a Drupal site to be exported into code and reused on other sites.

- An exportable configuration object, often called a feature *component*, is uniquely identified by a *machine-*

Some Additional Vocabulary

- A *dependency* is created when one module requires another in order to function. A dependency can be *hard* (when module A won't even install if module B isn't present) or *soft* (when module A will install but won't be fully functional without module B). Dependencies are an important consideration in distribution authoring because they can lead to incompatibilities between one set of features and another.

- A *patch* is a text file containing a suggested code-level change to an existing piece of code, like a module or Drupal core.

- A *theme* is the set of templates that gives a site its particular look and feel.

readable name or (less frequently) a universally unique identifier (UUID).

- One or more features and their supporting dependencies can be packaged together into an *app* that then can be installed with the Apps module. Many but not all Drupal distributions are being built using the apps model.

- A *base distribution* is a distribution that's specifically written to provide commonly needed functionality for use in other distributions.

Distributions and Interoperability

Interoperability is the ability of different systems to work together. When we talk about interoperability and Drupal distributions we mean: how can we make sure the building blocks of different distributions fit together in consistent and understandable ways?

The Origin of Features Much of the toolset for building distributions began with Open Atrium, a Drupal distribution that provides an advanced, team-based project management solution. Open Atrium devs took the time to build out a framework useful not just for their own product but for any distribution. The result was the Features module and a host of supporting add-ons.

Features quickly caught on with Drupal distribution developers. But in using it, they hit a number of snags.

Written on a one-off basis, each distribution had its own particular logic and structure. If you wanted to extend Open Atrium, you had to wade in and learn a lot about how that distribution was put together. And if you loved a particular piece of Open Atrium (say, its event calendar) and wanted to pull just that feature into your own site—well, good luck. Each feature was highly dependent on a whole set of assumptions and dependencies specific to the distribution.

Disentangling the Web looked like way more work than it was worth.

Kit The Open Atrium devs quickly recognized these issues and dug into addressing them. The result was Kit, a specification for feature development. The Kit Feature Specification sets out guidelines that feature developers can follow to help ensure that different features integrate seamlessly together, no matter what distribution or purpose they were built for.

What's in Kit? The good news is mainly some pretty sensible and easy-to-follow naming conventions. Kit answers questions like: how should you name the paths to your main feature landing pages? By following Kit, you'll help ensure that, if your work is combined with other Kit-compliant features, the result won't be

Why Build Interoperable Features, Apps and Distributions?

You're already taking considerable time and effort to build a distribution. Why put in the extra work to make it interoperable?

- *You'll be able to integrate other features or apps.* There's no need to go it alone. By building your distribution with interoperability foremost in mind, you'll be able to add in what others already have produced and be able to focus on just the key pieces that interest you.

- *More distros and sites will be able to use your work.* It's frustrating to build something and then watch someone else reproduce pretty much exactly what you already did only with a few details changed. If you make it easy to do so, others will be more likely to use, and maybe even contribute back to, what you've begun.

a mishmash of disparate configuration.

But, there's a lot that didn't make it into Kit. Here are some practical steps beyond Kit to help ensure that your work integrates seamlessly with other distributions.

Practical Steps in Building Interoperable Distributions

So far, I've been describing distribution building in pretty general terms. Now it's time to jump into some practical problems and real-world examples. If you're a developer relatively new to Drupal and some of this doesn't immediately make a lot of intrinsic sense, don't worry—it'll all be a lot clearer after you roll up your sleeves and dig in a bit. And if code blocks make your eyes swim, just pretend they aren't there—you can do a lot without touching a line of code.

Study Up on Existing Distributions A first good step is to get up close and comfortable with some of the existing distributions. You might choose to build your features on the same model as those of an existing distribution or feature set, or even directly add to an existing feature set.

For example, the Debut feature set includes a Debut Feature Specification that extends and updates Kit, providing guidelines for a higher level of integration. The Debut project page on drupal.org also links to developer documentation on building new Debut features.

Other distributions like OpenPublish include detailed documentation on their Web sites about how to write features or apps that extend the distribution. Read up!

Consider Building Off a Base Distribution Panopoly is the first distribution written specifically to be a base for other more-specific distributions. Panopoly tackles some key administrative, usability and design challenges that any distros will face. Building a distribution off Panopoly will save you having to solve all those problems yourself. It also will facilitate interoperability—at least with other Panopoly-based distros. The Panopoly project page on drupal.org includes documentation on how to base a distribution off of Panopoly.

Spark is a newer initiative that's also tackling usability and design challenges. It's not yet fully primed to be a base for building other distros, but discussions are underway, and it's a project to keep an eye on.

Consider Packaging Your Work as Apps Not everyone is thrilled with the Apps model. Apps makes respected Drupal developer and evangelist Angie Byron (webchick) wince—judging by her comments in a recent thread on drupal.org. Her concerns, to paraphrase, were that the apps model clouds waters already muddied by features and modules and whatnot and also detracts from getting usability fixes into Drupal core and drupal.org where they can help everybody.

That said, well, Apps is being adopted as a tool in many Drupal 7 distributions and can help solve both development and usability issues.

For developers, Apps makes it easier to incorporate features from other distros into their own. For example, Apps is what facilitates using Panopoly as the basis of your own distribution.

For site builders, Apps gives a one-click solution for installing a feature—complete with all of its dependencies—on a site that wasn't based on a distribution.

Don't Re-invent the Feature Even beyond building off a base distribution, you may be able just to take and use a lot of what you need from existing features. Before writing yet another event or blog or mapping or social-media feature, take a good look at what's already out there.

Seriously ask yourself: what would I need to do to use the existing feature directly in my distribution? Incorporating other people's work can free you up to focus on what you're really interested in. Think of it this way:

the alternative is having to build and maintain *everything* yourself. Ouch!

And, what better path to compatibility could there be than actually using the same code?

Avoid Dependencies of One Feature on Another No, really, avoid them—even "soft" dependencies. Unless your new feature is strictly enhancing an existing feature—like, say, an "Event registration" feature that enhances an existing "Event" feature—having dependencies on other features is trouble.

The key problem with dependencies is that they quickly produce incompatibilities between one set of features and another. The Features module won't let you enable two different features that have one or more identically named components of the same type. You can avoid dependencies through careful planning of how you'll handle the components that are needed in more than one feature.

Keep Generic Components Out of Features Exportables Take, for example, a user role like "administrator" or "editor". These are generic components—they're going to be needed in many different features, so that each feature can assign appropriate permissions for tasks like creating a particular type of content.

Instead of exporting user roles to features, consider having each of your feature modules create the general-use roles it needs at install time using regular Drupal APIs. Here's an example of how you might do this in a module's install file using hook_install(), which Drupal core calls when a module is first installed:

```
/**
 * Implements hook_install().
 *
 * Create "administrator" and "editor" user roles
 * if they don't already exist.
 */
function example_install() {
  $role_names = array('administrator', 'editor');
  foreach ($role_names as $name) {
    // If there isn't an existing role by this
    // name, create one.
    if (!$role = user_role_load_by_name($name)) {
      $role = new StdClass();
      $role->name = $name;
      user_role_save($role);
    }
  }
}
```

And, even before you write your own custom install hooks, have a look at the Apps compatible module, which provides built-in support for handling shared components like user roles and taxonomy vocabularies.

Enhance Other Features without Requiring Them Say you're writing a feature that adds social-networking links to different content types. Great—but which content types should get the links?

If you hard-code content types from specific features into your social linking feature, you've introduced either soft or hard dependencies—and precluded other feature authors from using the same approach. Instead, you can build in a flexible responder that enables other features to register their content types for social linking easily. Here's an example.

Say you're building a feature that uses the Service links module, which stores configuration in a variable called "service_links_node_types". In your feature, include implementations of two Drupal code hooks that allow you to respond when modules are enabled.

Here's some pseudo code; see the Debut Social feature for a full working example:

```
/**
 * Implements hook_modules_enabled().
 *
 * When a module is enabled, check to see if it is
 * a feature module with a node type designated for
 * social linking.
 */
function example_modules_enabled($modules) {
  $social_types = array();

  // Load info from feature modules and filter to
  // get only the ones that were just installed.
  $features = features_get_features();
  $features = array_intersect_key(
    $features,
    drupal_map_assoc($modules)
  );

  // Iterate through the features.
  foreach ($features as $feature) {
    // Look at the $feature->info array for node
    // types and an 'example_social_types' key.
    // Add any hits to the $social_types array.
  }
  if (!empty($social_types)) {
    // Merge into the existing variable's value so
    // as not to overwrite existing settings, using
    // variable_get(), array_merge(), and then
    // variable_set().
  }
}

/**
 * Implements hook_enable().
 *
 * When this module is enabled, process any social
 * linking designations for all currently enabled
 * modules.
 */
function example_enable() {
  example_modules_installed(module_list());
}
```

Now, to get its content type recognized for social linking, a feature will need only a single line in its .info file:

```
example_social_types[example_node_type] = example_node_type
```

Distinguish between "Hard" and "Soft" Configuration Features provide reliable and consistent methods for capturing, reproducing and updating configuration. That's exactly what you want with what's sometimes called "hard" configuration: the settings and objects that are core to your feature or distribution and shouldn't change in different environments—stuff like content types and fields.

But there's also "soft" configuration—the stuff you might want to set initially but that site administrators can and should change on each site. Soft configuration includes things like the default theme used on the site. You want to set an initial default theme, but any site should be free to replace that with its own theme choice.

To set soft configuration, you can skip Features and instead use the regular Drupal API functions like theme_enable() and variable_set().

Choose Your Layout Solution and Stick with It One of the biggest divides between one distribution and another is layout—how are blocks of content arranged on the page? Here, there are two main options used in distros, each with their myriad advocates.

The Context module enables finely tuned control over block placement on any site page or section into any region that the site's theme provides.

The Panels module enables even more options for selecting what content to place, but places content into its own custom layouts and regions instead of those provided by the site's theme.

If Context was the early champion, the balance may be shifting to Panels, which appears to be closer to the layout solutions that will go into Drupal 8 core. The Drupal Commons distro, for example, is switching from Context to Panels with its Drupal 7 upgrade.

One of the biggest divides between one distribution and another is layout—how are blocks of content arranged on the page?

One approach that's used is to pull layout and other user interface stuff into a separate feature. That way, site builders or distro authors can choose to use the core of a feature even if they choose a different layout solution. It's an interesting idea but one that adds some extra layers of complexity to the feature-building process.

Which to choose is going to be is up to you. But the layout model you decide on will have a big impact on how your feature or distro fits with the crowd.

Keep It Stable, Proven and Lean There's a place for trying out the latest experimental alpha release of a cool module that someone posted last week, but it's probably not in your distro.

Stick wherever you can with the tried-and-true solutions that any skilled and experienced developer would choose for the purpose. Use recommended, stable releases rather than development versions. If you don't absolutely have to add a new dependency, don't. And avoid patches like the plague; they're bound to break if a site already has installed a non-patched version of the code.

Yes, there are exceptions to all of these rules. You might, on rare occasion, see no choice but to use a development version of a module or judiciously add a patch to address a critical bug. But every time you do, keep in mind you're cutting yourself off from other distros or features that might not have climbed out on the limb you're clinging to.

Save the Truly Specific for Your Install Profile Here's an example from the Open Outreach distribution. Open Outreach is built for nonprofit organizations, but it's based on Debut features, which are written to be generic enough to be used on many different types of sites.

One such feature is for event handling. Debut Event includes an "Event type" vocabulary that allows content editors to classify events by category, with events of each category getting a different color on the event calendar.

But sticking terms that are specific to the nonprofit organization use case directly into Debut Event would break the whole idea of a generic feature. So instead we put just that bit of enhancement into the Open Outreach install profile, using one of the many code hooks that Drupal provides for responding to data transactions on the site. Here's code adapted from openoutreach.profile:

```
/**
 * Implements hook_modules_installed().
 *
 * Add custom taxonomy terms to the event_type
 * vocabulary if it is created.
 */
function example_entity_insert($entity, $type) {
  if ($type == 'taxonomy_vocabulary' &&
    $entity->machine_name == 'event_type') {
    $names = array(
      'Conference',
      'Meeting',
      'Workshop'
    );
    foreach ($names as $name) {
      $term = new StdClass();
      $term->name = $name;
      $term->vid = $entity->vid;
      $term->vocabulary_machine_name =
        $entity->machine_name;
      taxonomy_term_save($term);
    }
  }
}
```

When the event type vocabulary is created, Open Outreach responds by adding a few initial event types that nonprofit organizations typically might use. The Debut Event feature doesn't have to know or care.

Distributions and Interoperability in Drupal 8

Version 8 of Drupal core is being developed through a series of initiatives, each focused on a distinct area of development and headed up by leaders in the field.

Probably the most sweeping Drupal 8 changes for distributions are coming out of the configuration management initiative (CMI). This initiative has given Drupal native support for file-based configuration. A lot of the workarounds that are necessary to

Resources

Tools for Working with Distributions:

Apps Module: **http://drupal.org/project/apps**

Apps Compatible Module: **http://drupal.org/project/apps_compatible**

Debut Feature Specification: **http://drupal.org/project/debut**

Entity API Module: **http://drupal.org/project/entity**

Features Module: **http://drupal.org/project/features**

Kit Feature Specification: **http://drupal.org/project/kit**

Distributions Hosted on Drupal.org: **http://drupal.org/project/distributions**

Drupal 8 Configuration Management Initiative:
http://groups.drupal.org/build-systems-change-management/cmi

Selected Drupal Distributions:

Commerce Kickstart: **http://drupal.org/project/commerce_kickstart**

Conference Organizing Distribution (COD): **http://drupal.org/project/cod**

Drupal Commons: **http://drupal.org/project/commons**

Open Atrium: **http://openatrium.com**

Open Outreach: **http://drupal.org/project/openoutreach**

OpenPublic: **http://drupal.org/project/openpublic**

OpenPublish: **http://drupal.org/project/openpublish**

Panopoly: **http://drupal.org/project/panopoly**

Spark: **http://drupal.org/project/spark** convince Drupal to recognize code-based configuration in Drupal 7 may soon be a thing of the past.

Big changes of course bring their share of challenges. There's a lot that still needs to settle out before it'll be clear just how features, apps and distributions will translate to Drupal 8. But with core Drupal devs solidly addressing configuration management in Drupal core, the future of Drupal distributions looks bright.

In the meantime, if you're venturing into the world of apps or features or distros, take the time to do it right. Draw on the best of what's out there. And, in turn, make your feature, app or distribution as interoperable as it can be. Doing so will help others make the most of your contribution and also help future-proof your work, whatever Drupal 8 brings.■

Nedjo Rogers has been an active Drupal contributor since he posted his first module in 2003. He is the technical lead of the Open Outreach Drupal distribution for nonprofit organizations and a partner at Chocolate Lily Web Projects, http://chocolatelilyweb.ca. When not coding, he writes poetry and folk songs and tries to learn the accordion.

Cache in Drupal 7

Drupal is a powerful and very flexible, but often heavy, platform. In this article, I show how to utilize all of its advantages and still make it perform fast.

JANEZ UREVC

Performance is one of the most important factors that should be considered when building Web sites. Users expect sites to be good-looking, feature-rich, loaded with multimedia and interactive. To achieve all these things, you often need to build heavy Web applications, which are not always performant by their nature. To achieve the best balance between features and performance, there are various techniques to use, and caching is one of the most popular.

It is relatively simple to build a small personal Web site or blog that will perform well. Like most other Web platforms, Drupal also provides built-in mechanisms that will allow you to achieve sufficient performance for such a site in a relatively simple way. However, things become much more interesting when you have to build a site with a few hundred-thousand content items and tens of millions of visits per month. The solution definitely will not be straightforward, but it can be done. Drupal and some of its contrib modules provide mechanisms that are powerful and flexible enough to build sites of this size successfully. I describe some of them here.

The solutions and techniques discussed in this article were tested while building a relatively large media site, which was migrated to Drupal 7 in December 2012. It currently is the biggest Drupal site in the wider Balkan region.

What Is Cache?

Cache is a frequently used technique in computing, for both hardware and software. The basic idea behind cache is to identify data that is used often and store it on a medium where it can be fetched as quickly as possible. When building Web applications, you usually want to load as much data as possible from memory (which is fast). On the other hand, you mostly need permanent storage for it. Since RAM is not such a medium, you have to keep the data on disk, which is very slow in most cases. This leads to a typical architecture for Web applications that uses disk to store data permanently and RAM as an efficient cache medium for the most-used pieces of it.

There are many cases when you usually will want to use cache in Web development, including the following:

- Heavy database queries.
- Output that is "expensive" to render.
- Complex numerical calculations.
- Code with substantial use of I/O (disk).
- Data that was fetched from remote servers.

Two caches should be considered by every PHP+MySQL developer:

- APC (Alternative PHP Cache) is a PHP extension that implements op-code cache. Op-code cache saves overhead that is caused by the fact that PHP compiles its scripts on every request. Because most of the scripts don't change often, APC saves time by saving compiled versions of scripts into shared memory to be available for the next requests.
- MySQL query cache saves query results to be available the next time the same query is executed. Substantial savings can be achieved this way, because repeating queries happens a lot in Web apps.

Cache in Drupal 7

Cache is implemented as a very flexible and pluggable framework in Drupal 7. Here are three of the most important concepts to understand:

- A cache item is a complete piece of data that is stored in cache. A developer addresses each cache item with a key that obviously should be unique.
- Cache bin is a group of similar cache items. There are a lot of different cache bins, as Drupal allows its modules to create their own. Each cache item generally can be saved to any bin available in the system.
- Cache back end is a technology platform that is used to store cache. Each cache bin generally can be stored in a different back end. There is support for numerous back ends in Drupal's contrib ecosystem (such as APC, Memcached, Redis and so on). It is not hard to implement support for a new back end if your project requires something that has not been supported yet.

If you develop a Drupal module and you want to utilize the advantages of Drupal's cache framework, you should do something like this:

```
<?php
if ($cache = cache_get('my_cache_item', 'cache')) {
  $data = $cache->data;
}
else {
  // some heavy calculations...

  cache_set('my_cache_item', $data, 'cache');
}

// Use $data
?>
```

The cache item in the above snippet is keyed with `my_cache_item` and stored in a bin called `cache`, which is obviously Drupal's default bin. Check Drupal's `cache_set()` (**http://dgo.to/a/cache_set**) and `cache_get()` (**http://dgo.to/a/cache_get**) functions if you need more information.

The following is a list of cache bins provided by Drupal core:

- cache — default (generic) cache bin.
- cache_block — storage for cached versions of built (rendered) blocks.
- cache_bootstrap — bin for data required to bootstrap Drupal.
- cache_field — storage for cached versions of Drupal's fields.
- cache_filter — bin for the Filter module to store already filtered pieces of text.
- cache_form — bin for the form system to store recently built forms and their storage data to be used in subsequent page requests.
- cache_image — bin used to store information about image manipulations that are in progress.
- cache_menu — storage for the menu system to store router information and menu link trees.
- cache_page — bin used to store compressed pages for anonymous users.
- cache_path — bin for path alias lookup.
- cache_update — bin for the Update module to store information about available releases, fetched from the central server.

Cache Back Ends

A lot of different cache back ends can be used together with Drupal. I describe some of the most popular ones here that also have support in at least one Drupal module. Back ends that are most commonly used in the Drupal ecosystem also are very popular in general Web development. Each of them has its own advantages and disadvantages, and each will satisfy different needs.

Some modules implement more than just cache support. They often implement sessions and semaphore (locking) handlers too. That also will speed up your site, as a typical Drupal Web site uses those mechanisms frequently. Refer to each module's project page and README.txt file in order to utilize their advantages.

Database Cache (Default)

Drupal's default cache implementation is obviously very easy to use, as you get it "for free" when you install Drupal. It also doesn't add any additional complexity to your server stack, because it uses a database (usually MySQL). This back end should work totally fine for most smaller sites. The problem is its speed, because it writes all data directly to disk. Writes are especially very slow, as MySQL's query cache helps to boost read performance. Another disadvantage is the fact that it uses a database that already is under very heavy load on high-traffic Web sites.

APC User Cache

I already mentioned APC (**http://drupal.org/project/apc**). It provides a lot of performance improvement, as PHP files generally do not change much. It is relatively simple to install and very easy to configure. It should be used on *every* Web server that runs PHP (production and development).

Another feature of APC that people are not always aware of is the *user cache*. The user cache uses APC's shared memory to store users' (developers') custom data, and it definitely can be used for cache purposes. It is *very* fast, because it stores data directly in PHP's memory. Assuming that you already have APC installed on your server (and you should), it also does not add further complexity to the server stack. Data in APC is not stored permanently; you will lose all your data if the power goes down. Another disadvantage is the fact that each server maintains its own version of cache. This will cause cache warm-up to take longer in high-availability (more Web servers) setups. This also can lead to synchronization problems.

Memcached

Memcached is an open-source, high-performance, distributed memory object caching system (**http://memcached.org**). It is very popular and is used on many well-known sites. It runs as a separate dæmon, which means that it adds another level of complexity to the server stack. It is very easy to configure and administer, so

NO
SPEED
LIMIT

New Relic®
SPEED UP YOUR DRUPAL DEVELOPMENT
Improve and monitor site speed
Monitor, manage and improve the end-to-end performance of your web apps with one simple tool.

that shouldn't be a problem. Applications communicate with it via the network (TCP or UDP). You will have to install another PHP extension in order to use it with PHP applications. It stores all data in memory, which makes the data non-permanent. All Web servers will use the same Memcached pool in high-availability environments, which is another big advantage. See **http://drupal.org/project/memcache** for more information.

Redis

Redis is an open-source, advanced key-value store (**http://redis.io**). It is very similar to Memcached. It is fast, centralized and relatively easy to configure. It stores your data in memory, but it sooner or later writes it to disk, which makes it permanent. Write frequency can be configured, which gives you the power to balance between performance and security of your data. It needs a PHP extension, just as Memcached does. It also comes with a PHP library. Redis is very fast, but sooner or later it will need access to I/O, which could cause some performance overhead in environments that already are under heavy load. See also **http://drupal.org/project/redis**.

MongoDB

MongoDB is a scalable, high-performance, open-source NoSQL (document-oriented) database (**http://www.mongodb.org**). It

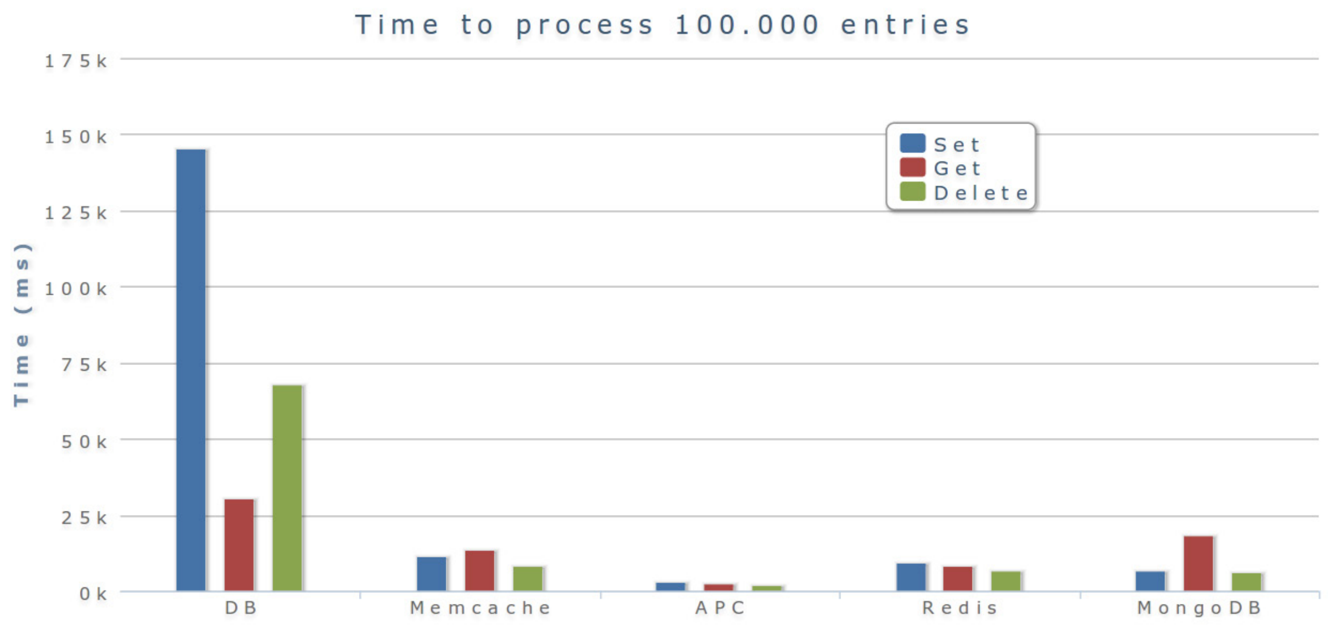


Figure 1. This performance overview of all the back ends described in this article shows that APC is the fastest and MySQL (DB) the slowest. Redis and Memcached are relatively similar when it comes to performance, but that could change if Redis would write its data to disk often. MongoDB is very fast, especially when it comes to writes.

is much faster than MySQL, and it stores data permanently. You will need to install a PHP extension in order to use it. It is probably the most complex to configure and administer of all the back ends I describe here. MongoDB's other advantage is its ability to store Drupal's content, which makes for a really powerful content database in addition to a powerful cache back end.

How to Configure Cache Back Ends

You will need to do some configuration in Drupal's settings.php in order to use different cache back ends. First, you need to include cache back-end implementations:

```
# Memcache
include_once('./includes/cache.inc');
include_once('./sites/all/modules/memcache/memcache.inc');
$conf['memcache_key_prefix'] = 'drupal';

# Redis
$conf['cache_backends'][] =
 ➥'sites/all/modules/redis/redis.autoload.inc';
$conf['redis_client_interface'] = 'PhpRedis';  //Library or extension

# APC
$conf['cache_backends'][] =
 ➥'sites/all/modules/apc/drupal_apc_cache.inc';

# Mongo
$conf['cache_backends'][] =
 ➥'sites/all/modules/mongodb/mongodb_cache/mongodb_cache.inc';
```

The paths in the above example should be adjusted to your specific installation. When you include cache back ends, you need to set the default:

```
$conf['cache_default_class'] = 'MemCacheDrupal';
//$conf['cache_default_class'] = 'Redis_Cache';
//$conf['cache_default_class'] = 'DrupalAPCCache';
//$conf['cache_default_class'] = 'DrupalMongoDBCache';
//$conf['cache_default_class'] = 'DrupalDatabaseCache';
```

In the above example, Memcached is configured as a default cache back end. The appropriate line should be uncommented in order to select a different option. Besides the default back end, where all bins will go, you also can define per-bin back ends:

```
// cache_form bin
$conf['cache_cache_form'] = 'Redis_Cache';

// cache_menu bin
$conf['cache_cache_menu'] = 'DrupalMongoDBCache';

// etc....
```

In the above example, the bin `cache_form` is configured to go into Redis, and the bin `cache_menu` is set to be stored in MongoDB.

Cache Implementations in Drupal

Drupal core and some of its modules utilize the cache framework to implement user-friendly, easy-to-configure caching.

Page and Block Cache

Page and block cache are implemented in Drupal core, and you'll get them with every Drupal installation. Page cache will save the HTML output of the entire page for anonymous users and store it to the `cache_page` bin. This HTML will be displayed to all anonymous users, which will bypass most of the Drupal bootstrap and the entire page-generation process. This will save a huge amount of time and server resources.

Block cache, on the other hand, will save HTML output of Drupal blocks (parts of a page). Blocks also can be cached for registered users, which will give you some performance improvement for non-anonymous traffic. Blocks are stored in the `cache_block` bin.

Both are very easy to enable and configure by navigating to admin/config/devel/performance.

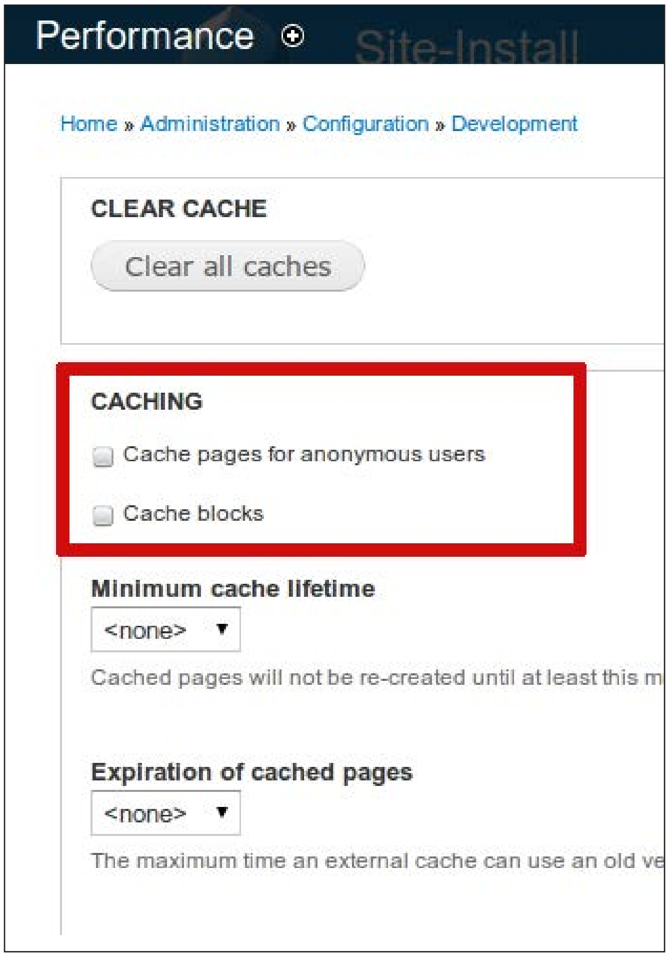
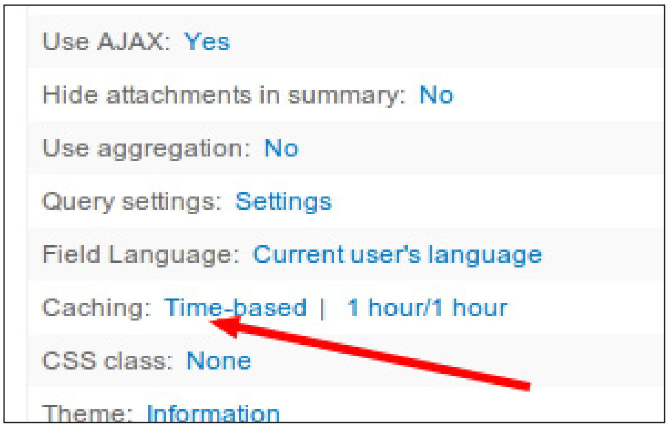

Performance
Site-Install
Home » Administration » Configuration » Development
CLEAR CACHE
Clear all caches
CACHING
Cache pages for anonymous users
Cache blocks
Minimum cache lifetime
<none>
Cached pages will not be re-created until at least this m
Expiration of cached pages
<none>
The maximum time an external cache can use an old ve

Figure 2. Configuration for Page and Block Cache

Views Cache

Views is the most frequently used Drupal module (**http://drupal.org/project/views**). It provides a flexible method for Drupal site designers to control how lists and tables of content, users, taxonomy terms and other data are presented. There probably aren't many serious Drupal Web sites that don't use this module.

Views implement a cache mechanism that will save the result of a view's query and its HTML output. Views will operate with its cache on a timely basis, but you can implement your own plugin to achieve totally customized behavior. The



Use AJAX: Yes
Hide attachments in summary: No
Use aggregation: No
Query settings: Settings
Field Language: Current user's language
Caching: Time-based | 1 hour/1 hour
CSS class: None
Theme: Information

Figure 3. Configuration for the Views Cache

views cache can be configured per every view, which gives you a lot of flexibility. Cache items will be stored in the `views_cache_data` bin.

Usage of the views cache should be considered as an option on every Drupal Web site that uses the Views module.

Panels Cache

Panels is another powerful Drupal module (**http://drupal.org/project/panels**). It allows you to build complex page layouts and configure and control them in a really intuitive way. Besides its many other features, it also has a powerful cache implementation. Panels, by default,

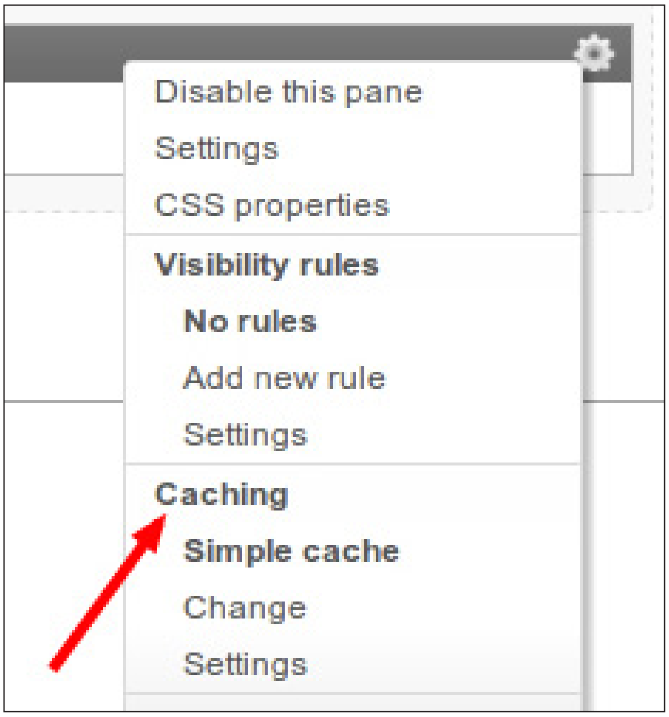
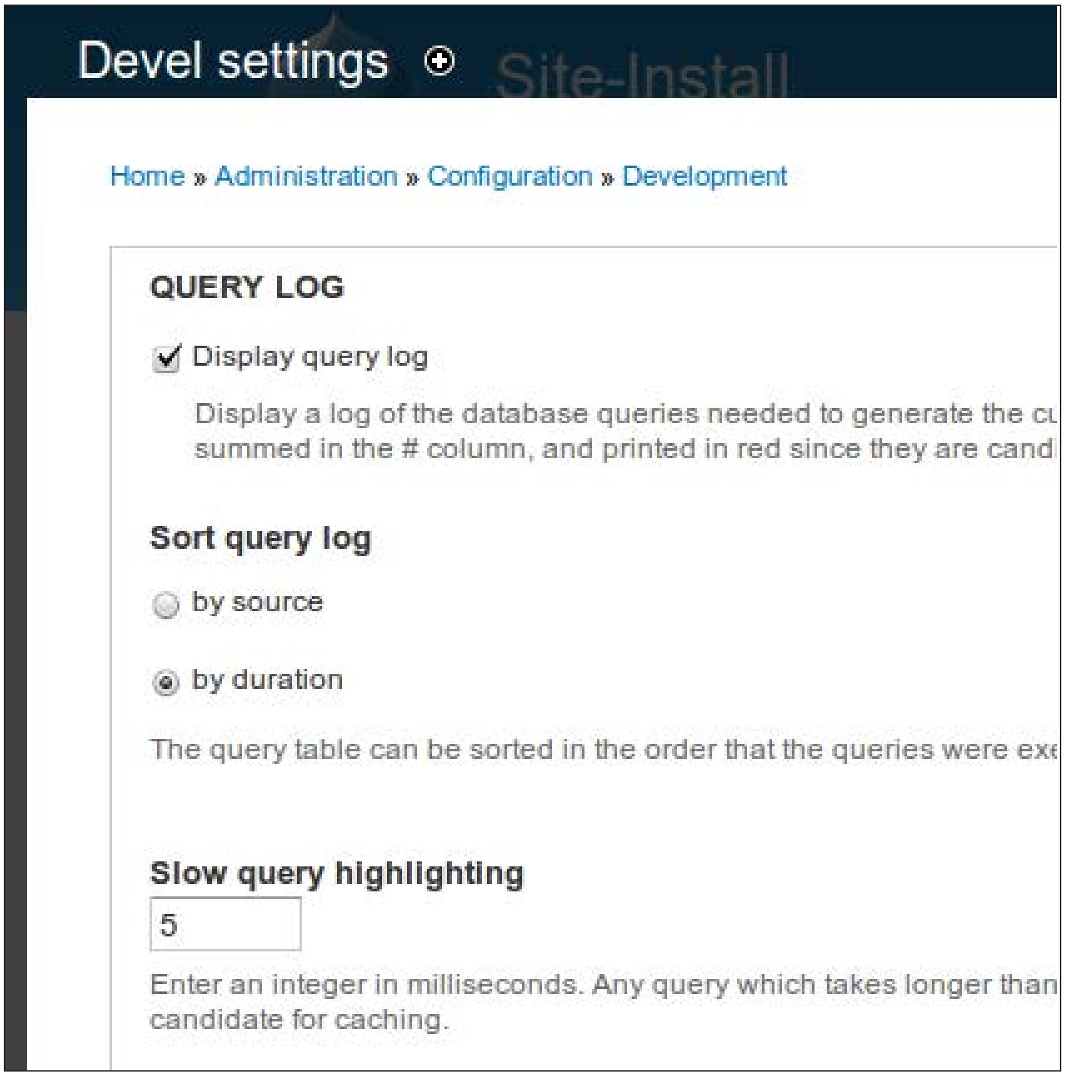


Figure 4. Configuration for the Panels Cache



Devel settings ⊕ Site-Install

Home » Administration » Configuration » Development

QUERY LOG

[✓] Display query log

Display a log of the database queries needed to generate the cu
summed in the # column, and printed in red since they are cand

Sort query log

○ by source

● by duration

The query table can be sorted in the order that the queries were exe

Slow query highlighting

5

Enter an integer in milliseconds. Any query which takes longer than
candidate for caching.

Figure 5. Configuration of Devel's Query Cache

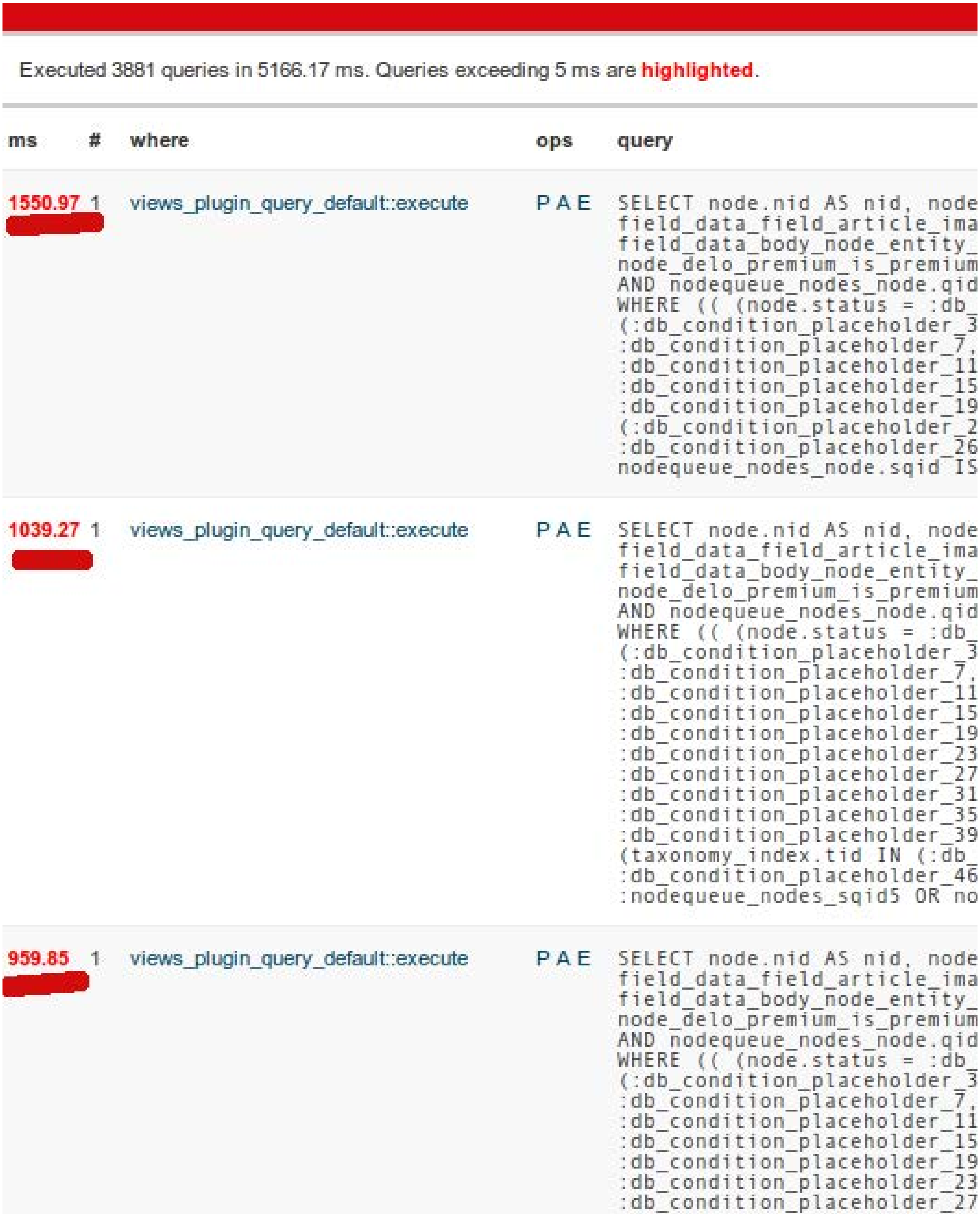
will allow you to cache almost everything and configure granulation and lifetime. Again, this can be completely customized with your own custom plugins, resulting in even more flexibility.

The panels cache is configured on the configuration page for each panel. It is usually configured on a per-pane (part of a page) basis.

How to Optimize Your Drupal Page for Performance

Cache is a very powerful tool in Drupal, but it won't help you much if you don't optimize your queries and code first. Cache will just save you from running them over and over again, but you still will have to execute your slow queries and run your slow code from time to time. However, Drupal provides a lot of tools to optimize those things too.

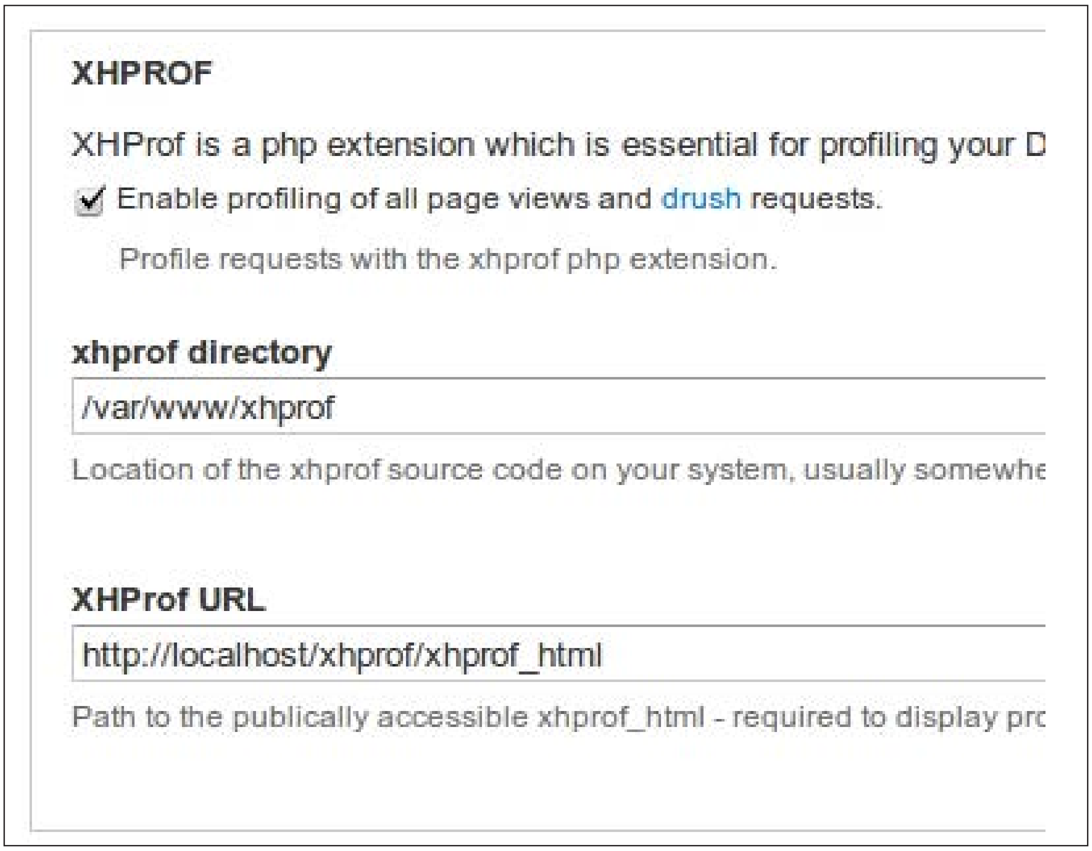
The Devel module (**http://drupal.org/project/devel**) implements a handy query log, which will print the list of all queries that were executed during a given request in the footer of your page. This allows



Executed 3881 queries in 5166.17 ms. Queries exceeding 5 ms are **highlighted**.

```
ms       #   where                                     ops   query

1550.97  1   views_plugin_query_default::execute       P A E SELECT node.nid AS nid, node
                                                              field_data_field_article_ima
                                                              field_data_body_node_entity_
                                                              node_delo_premium_is_premium
                                                              AND nodequeue_nodes_node.qid
                                                              WHERE (( (node.status = :db_
                                                              (:db_condition_placeholder_3
                                                              :db_condition_placeholder_7,
                                                              :db_condition_placeholder_11
                                                              :db_condition_placeholder_15
                                                              :db_condition_placeholder_19
                                                              (:db_condition_placeholder_2
                                                              :db_condition_placeholder_26
                                                              nodequeue_nodes_node.sqid IS

1039.27  1   views_plugin_query_default::execute       P A E SELECT node.nid AS nid, node
                                                              field_data_field_article_ima
                                                              field_data_body_node_entity_
                                                              node_delo_premium_is_premium
                                                              AND nodequeue_nodes_node.qid
                                                              WHERE (( (node.status = :db_
                                                              (:db_condition_placeholder_3
                                                              :db_condition_placeholder_7,
                                                              :db_condition_placeholder_11
                                                              :db_condition_placeholder_15
                                                              :db_condition_placeholder_19
                                                              :db_condition_placeholder_23
                                                              :db_condition_placeholder_27
                                                              :db_condition_placeholder_31
                                                              :db_condition_placeholder_35
                                                              :db_condition_placeholder_39
                                                              (taxonomy_index.tid IN (:db_
                                                              :db_condition_placeholder_46
                                                              :nodequeue_nodes_sqid5 OR no

959.85   1   views_plugin_query_default::execute       P A E SELECT node.nid AS nid, node
                                                              field_data_field_article_ima
                                                              field_data_body_node_entity_
                                                              node_delo_premium_is_premium
                                                              AND nodequeue_nodes_node.qid
                                                              WHERE (( (node.status = :db_
                                                              (:db_condition_placeholder_3
                                                              :db_condition_placeholder_7,
                                                              :db_condition_placeholder_11
                                                              :db_condition_placeholder_15
                                                              :db_condition_placeholder_19
                                                              :db_condition_placeholder_23
                                                              :db_condition_placeholder_27
```

Figure 6. Example of a Displayed Query Log (with Slow Queries)


XHPROF

XHProf is a php extension which is essential for profiling your D

✓ Enable profiling of all page views and drush requests.

Profile requests with the xhprof php extension.

xhprof directory

/var/www/xhprof

Location of the xhprof source code on your system, usually somewhe

XHProf URL

http://localhost/xhprof/xhprof_html

Path to the publically accessible xhprof_html - required to display prc

Figure 7. XHProf Profiler

you to identify slow database queries and optimize them. Doing some minor modifications of a query (view) or an index on a table will help you a lot in most cases.

Another handy tool is the profiler (**http://en.wikipedia.org/wiki/Profiling_(computer_programming)**). It will allow you to identify parts of your code that are slow or that use an abnormal amount of memory. When it comes to PHP, I recommend XHProf (**https://github.com/facebook/xhprof**), which is really easy to use. Once you install and configure it, the Devel module gives you Drupal integration for it. You just have to enable it, and Devel will run it automatically on every request and display a link to the profiling information at the bottom of the page.

Once you have optimized your code, you can start using cache. Identify parts of the page that are candidates for caching. When you have cached all parts of the page, enable page cache or use a reverse proxy. Reverse proxy does a similar job as a page cache, but it is a separate dæmon that sits in front of the Web server and is heavily optimized for this job. The most popular reverse proxy in the Drupal community is Varnish (**https://www.varnish-cache.org**), which also has a contrib module that integrates it with Drupal (**http://drupal.org/project/varnish**). Another module that does a similar job is Boost (**http://drupal.org/project/boost**). Boost will save the HTML output of entire pages to ordinary HTML files, allowing the Web server to serve static files for anonymous users.

Conclusion

Drupal has been known as a relatively slow platform, which is somewhat true. But, it also is very flexible and provides a lot of room for performance optimizations. This could make it even faster than its better-optimized alternatives in the end, while still preserving its strength and flexibility.

The performance optimization tools that come with Drupal core definitely will work for small and medium Web sites. When you start dealing with larger Web sites, you'll need to know your requirements and needs and use a more customized approach. You will, however, find a lot of tools and best practices even for large projects, which makes Drupal an excellent platform for almost every type of Web project.■

Janez Urevc is a Drupal developer from Slovenia, EU. He really loves the things he does, and that's why he feels that "every day is the best day of his life". He's been dedicated to open source since high school. He graduated in Software Development from the Computer and Information Sciences department at the University of Ljubljana. His Bachelor's thesis was focused on the implementation of Agile principles and Scrum methodology in a Web development department of a large media company. Besides Drupal, he's also passionate about almost everything connected to the Web, open source, Linux and software development. He participated in the 2011 Google Summer of Code, where he worked on the Media derivatives API for Drupal.

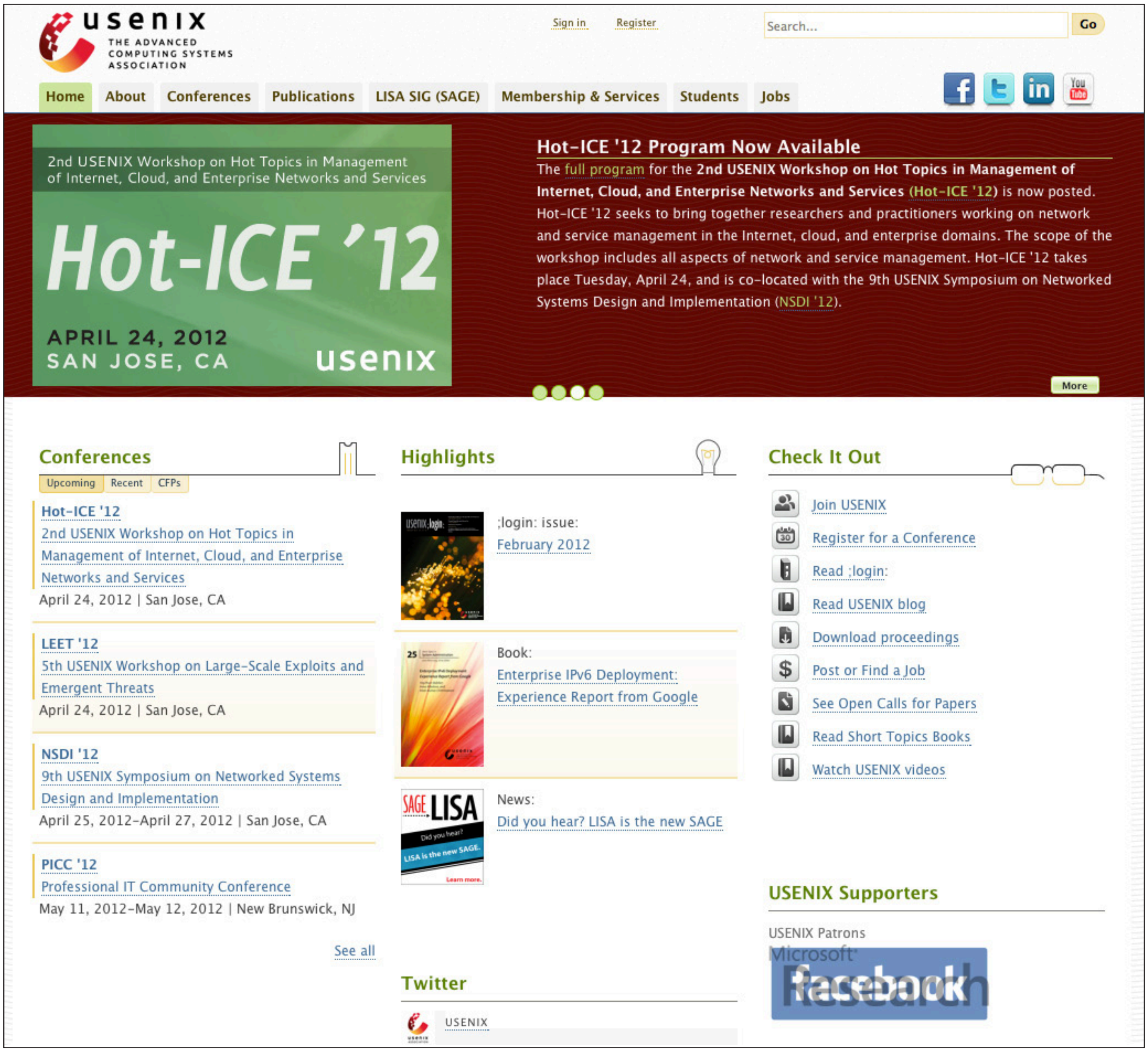
USENIX.org: a Case Study of a Migration to Drupal

The new USENIX.org is built on Drupal with an extensive Salesforce.com integration, Apache Solr-based faceted searching, e-commerce from the Drupal Commerce suite and microsites built with Organic Groups. Dozens of custom Drupal modules make this powerful system simple for staff and visitors.

JODY HAMILTON

USENIX (**http://usenix.org**), the Advanced Computing Systems Association, was founded in 1975 to foster and promote technical excellence and innovation among system administrators and other .nix professionals. This member-focused organization runs dozens of conferences each year, publishing content on-line for each event. Prior to the migration to Drupal, most conferences and Web pages were created using either straight HTML or PERL, and after 17 years of working with its Web properties in this way, USENIX had tens of thousands of custom HTML pages with a slew of Perl scripts and no database. The lack of a database-driven system meant it was never possible for the USENIX site to have much structural organization or coherence—it had simply grown organically for decades—and it created a management nightmare for those tasked with maintaining this growing library of content and functionality.

When we began the project, USENIX knew it needed a robust Content Management System (CMS) and had chosen Drupal. The new USENIX Web site requirements included:


usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Sign in Register

Search... Go

Home About Conferences Publications LISA SIG (SAGE) Membership & Services Students Jobs

2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services

Hot-ICE '12

APRIL 24, 2012
SAN JOSE, CA
usenix

Hot-ICE '12 Program Now Available

The full program for the **2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE '12)** is now posted. Hot-ICE '12 seeks to bring together researchers and practitioners working on network and service management in the Internet, cloud, and enterprise domains. The scope of the workshop includes all aspects of network and service management. Hot-ICE '12 takes place Tuesday, April 24, and is co-located with the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12).

More

Conferences

Upcoming Recent CFPs

Hot-ICE '12
2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services
April 24, 2012 | San Jose, CA

LEET '12
5th USENIX Workshop on Large-Scale Exploits and Emergent Threats
April 24, 2012 | San Jose, CA

NSDI '12
9th USENIX Symposium on Networked Systems Design and Implementation
April 25, 2012–April 27, 2012 | San Jose, CA

PICC '12
Professional IT Community Conference
May 11, 2012–May 12, 2012 | New Brunswick, NJ

See all

Highlights

;login: issue:
February 2012

Book:
Enterprise IPv6 Deployment: Experience Report from Google

News:
Did you hear? LISA is the new SAGE

Twitter

USENIX

Check It Out

Join USENIX
Register for a Conference
Read ;login:
Read USENIX blog
Download proceedings
Post or Find a Job
See Open Calls for Papers
Read Short Topics Books
Watch USENIX videos

USENIX Supporters

USENIX Patrons

Figure 1. The New USENIX.org

- Replacement of the old HTML and PERL script functionality.
- Strong security.
- Robust user and role management.
- Seamless integration with its newly adopted Customer Relationship Management system and its Event Management system (Salesforce.com and CVent, respectively).
- E-commerce functionality.
- Video integration.

- Private and public files.
- Faceted search listings.
- Migration of the data stored in the HTML pages to the new system.

Security, e-commerce, user/role management and faceted search were among the features best matching Drupal's strengths. USENIX also is interested in supporting open-source software communities, and wanted to develop its site in a way that gives back as much as possible. My team at Zivtech was chosen for the project due to our experience with Salesforce.com Drupal integrations, Ubercart customizations and community contributions.

Building in Drupal 6, Migrating to Drupal 7

When we started working with USENIX, Drupal 7 did not have a full release, so the site was built in Drupal 6. We used the Ubercart module suite for e-commerce features and Houston (**http://houstoncommand.com**) to handle the integration between Drupal and the Salesforce.com CRM, which also was under active construction.

Most of the USENIX Web site already was complete in Drupal 6 when a decision was made to upgrade to Drupal 7. This was very early in the Drupal 7 release cycle, so our team had to port 32 custom modules, 12 custom features, numerous contributed modules and our custom themes. Beyond these upgrades, being an early adopter meant that our developers had to commit dozens of additional patches to fix bugs and issues with upgrade paths from Drupal 6 to 7. During this process, we repeatedly fine-tuned our four-hour upgrade bash script, which consisted mainly of drush (the excellent Drupal shell interface) commands. In addition, we rebuilt the Web site's e-commerce functionality using Drupal Commerce, ported Houston to Drupal 7, and also re-created numerous views, organic groups configurations, profiles and more.

Upgrading a complex site to a new major version of Drupal is difficult because Drupal core (and to varying extents contributed modules) provides a migration path for data but no support for legacy code. Often major contributed building blocks of Drupal use a new core version as an opportunity for a rewrite, often without an upgrade path. Upgrades are more difficult if they occur sooner after the core release; if the site is live (as it typically is); and if the site database is updated frequently by visitors (purchases, account creation, comments and so on), and the site cannot be taken off-line for upgrade. With these trade-offs in mind, upgrading before site launch was preferable to postponing.

E-commerce

USENIX sells memberships, file access,

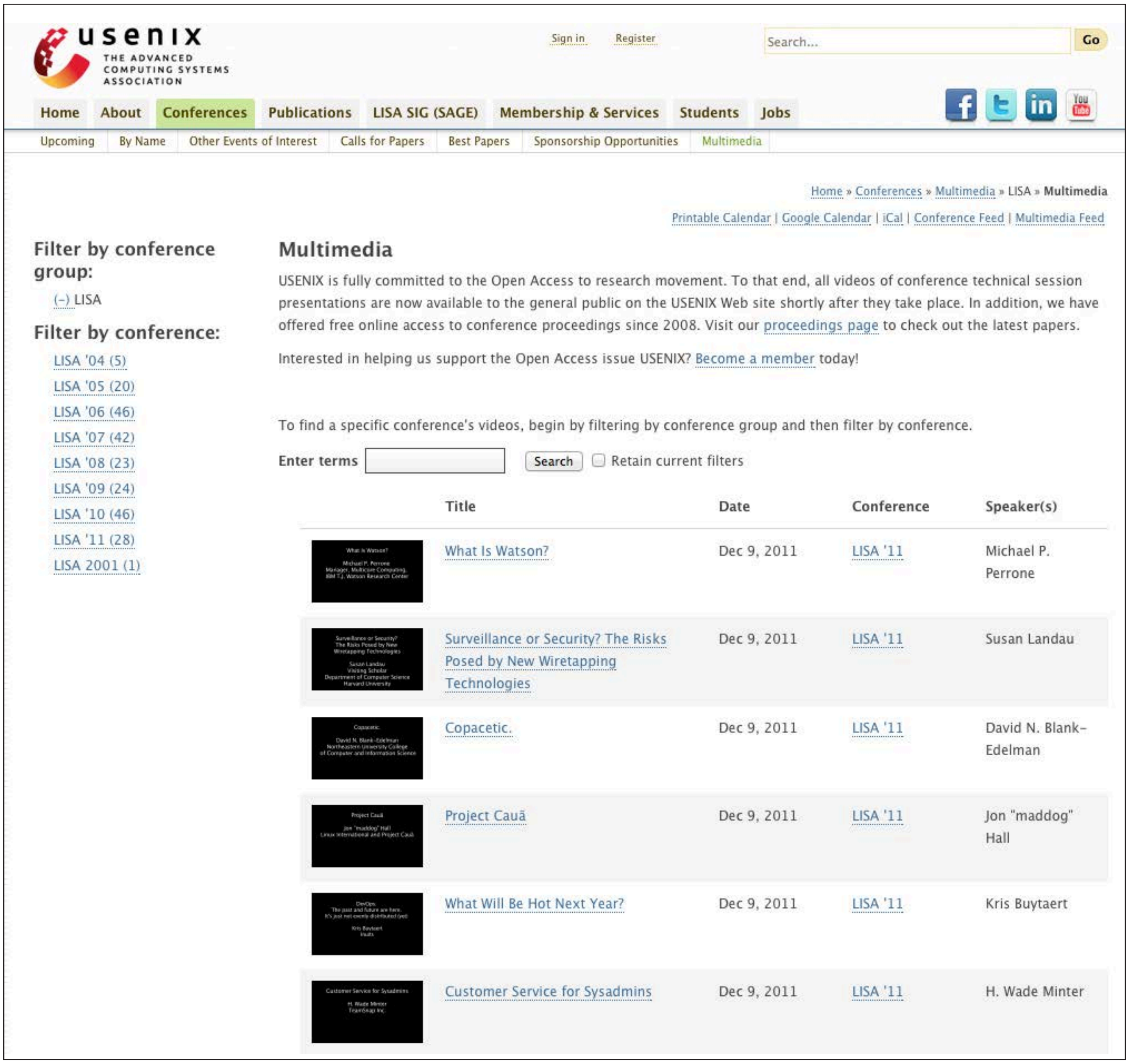
physical magazines, books and offprints. To cater to these diverse e-commerce needs, our team used the Drupal Commerce framework along with Commerce Coupon, Commerce Custom Line Items, Commerce Devel, Commerce Features, Commerce File, Commerce Flat Rate, Commerce Payflo Pro and Commerce Shipping to enable the sale, pricing and access to products.

Certain files on the Web site are free to select users (depending on membership status or conference registration) but unavailable or available for purchase by others. Files also can be set to become public at a specified date, so that they are no longer conceptually a product for sale on the site. Multiple product formats for one piece of content are routinely sold (such as print and electronic book formats). These needs did not map well administratively to the Drupal Commerce model of separating the product and its display (the content, aka Drupal node) and would have made for a complex and tedious workflow for content creation. Fortunately, the Commerce API facilitated a solution. Our team was able to hide the creation of products from Web site administrators by auto-creating and updating associated product files, SKUs, attributes and prices when content is created and updated. We also used the Commerce API to implement sophisticated membership levels and benefits, including free products per year of membership and discount scenarios. The Commerce API was a pleasure to use because it uses the same entity-based architecture as core Drupal 7.

Houston

USENIX required a complex bi-directional integration between Salesforce.com and the new Web site. We used our own Houston Command Center (**http://houstoncommand.com**) over the more widely used Salesforce Drupal module to allow flexibility with the data model as well as to provide queue management and failover support. Houston is OSS we've evolved during several large integration projects. It addresses challenges in integrating with SaaS products, including handling API limits and downtime. It provides a framework for mapping the data model between systems and handles queuing and backups.

Using Houston, we linked authentication, user profiles, membership status, conference registration, grant applications, pricebook entries and e-commerce purchases between the two platforms in a seamless manner. Existing USENIX.org members are able to log in to the new site with the same credentials they used pre-Drupal (which were mainly used in .htaccess-style authentications) by comparing their hashed passwords to those migrated into the CRM. Once authenticated, users can manage their membership and profile information on the new Drupal site, and USENIX staff can view and edit that data on Salesforce.com or on USENIX.org. When users register for a conference on the site, their profile information is used to pre-populate a registration form on CVent (an SaaS event management system). After payment, the CVent registration data is sent to the Salesforce.com CRM, where it is then available to Houston and the Drupal site to grant access to attendee-restricted pages and files. Staff even can manage the prices of different categories of products sold on the Drupal site (for example, the current price of an audiobook for a USENIX member) from the Salesforce.com CRM. By centralizing user history, such as



Home » Conferences » Multimedia » LISA » **Multimedia**

Printable Calendar | Google Calendar | iCal | Conference Feed | Multimedia Feed

Filter by conference group:

(-) LISA

Filter by conference:

LISA '04 (5)
LISA '05 (20)
LISA '06 (46)
LISA '07 (42)
LISA '08 (23)
LISA '09 (24)
LISA '10 (46)
LISA '11 (28)
LISA 2001 (1)

Multimedia

USENIX is fully committed to the Open Access to research movement. To that end, all videos of conference technical session presentations are now available to the general public on the USENIX Web site shortly after they take place. In addition, we have offered free online access to conference proceedings since 2008. Visit our proceedings page to check out the latest papers.

Interested in helping us support the Open Access issue USENIX? Become a member today!

To find a specific conference's videos, begin by filtering by conference group and then filter by conference.

Enter terms [ ] Search ☐ Retain current filters

Title	Date	Conference	Speaker(s)
What Is Watson?	Dec 9, 2011	LISA '11	Michael P. Perrone
Surveillance or Security? The Risks Posed by New Wiretapping Technologies	Dec 9, 2011	LISA '11	Susan Landau
Copacetic.	Dec 9, 2011	LISA '11	David N. Blank-Edelman
Project Cauã	Dec 9, 2011	LISA '11	Jon "maddog" Hall
What Will Be Hot Next Year?	Dec 9, 2011	LISA '11	Kris Buytaert
Customer Service for Sysadmins	Dec 9, 2011	LISA '11	H. Wade Minter

Figure 2. The Multimedia page is a searchable Solr-based listing of all content containing video or audio recordings, allowing faceting into specific conferences.

conference attendance, membership status, product purchases and profile information in the CRM, the organization can target its outreach and marketing efforts effectively.

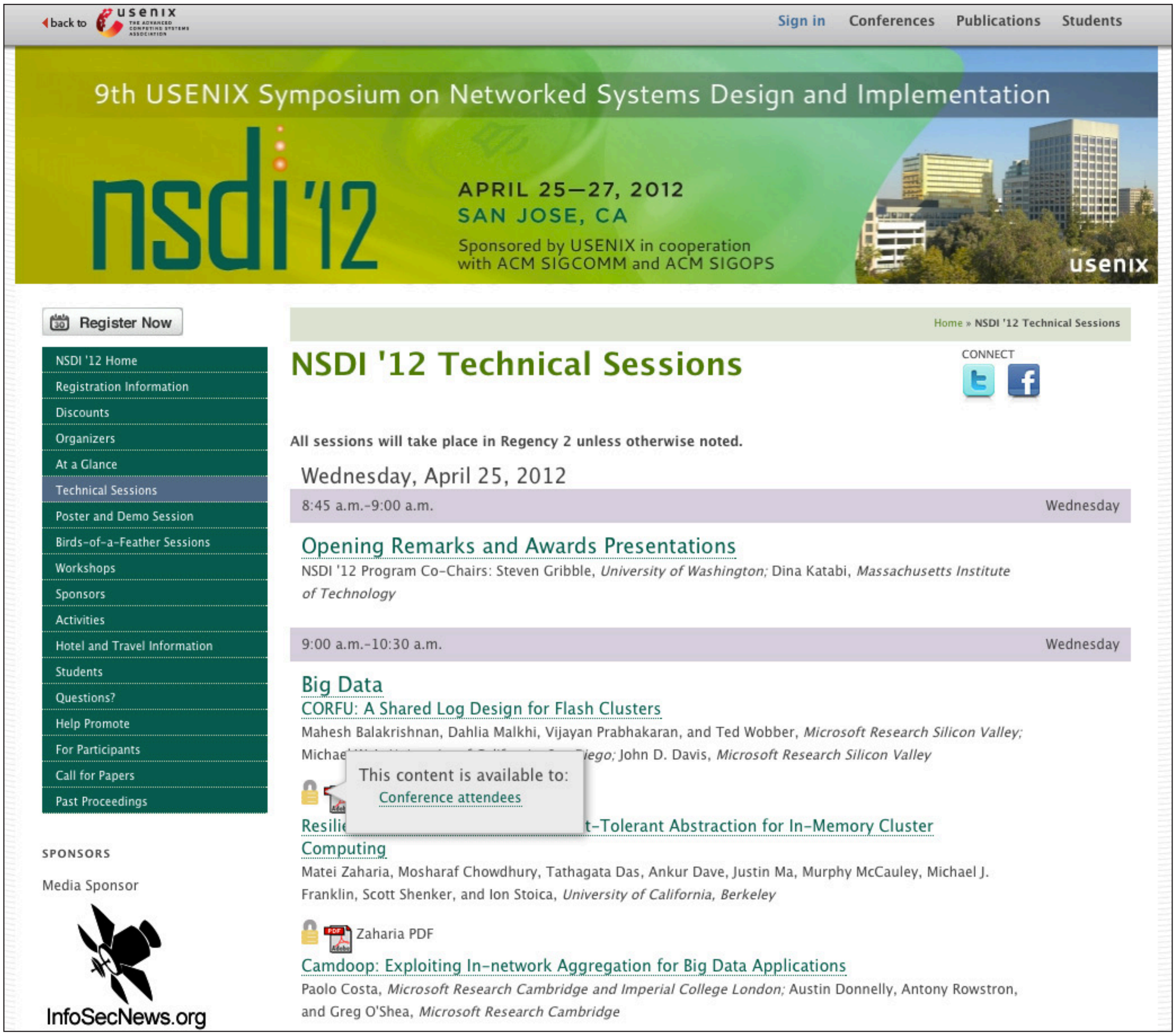
Search

We love using Apache Solr integrated with Drupal to build not only site-wide searches but also searchable, filterable listing pages. We build listings with Solr rather than traditional (usually Views module-based) database queries to get fast results, full content search and facets. We created three search pages, one each for proceedings, multimedia and site-wide search. Each uses custom Solr index fields, custom Facet API module sorts and custom search-result displays. We also created (and contributed) Facet API drop-down menus for the proceedings and multimedia search pages. Because the search work was built first for Drupal 6, we used the Apache Solr Search Integration module rather than the next-generation Search API Drupal suite. Search API allows site builders to use the Views module to build custom search listings quickly through a UI.

Microsites

Organic Groups, the Drupal module that powers groups.drupal.org to allow groups of people to organize themselves "organically", can be used effectively in many use cases that involve "groups" of site users and content. For USENIX.org, the "groups" are the conferences, and both users (attendees) and content (sessions, trainings, information pages) can be a part of a conference. To create a microsite for a conference in the past, USENIX staff copied a directory of HTML, CSS and scripts and modified its content and visual style for each conference. To allow a similar feel of distinct microsites, we used the Organic Groups Theme module to allow a unique Drupal theme to be assigned to any conference. The look of conference microsites is customized through the UI with banner logos, and the visual themes are created with CSS. These custom themes are subthemes of a conference base theme. We included a conference subtheme starter kit with basic CSS for common elements altered between conferences. The Organic Groups Menu module provides each conference with its own navigation. Each microsite consists of page, session, paper, training, sponsor, speaker and organizer content types.

The conference microsites display conference-specific Views module listings (such as lists of organizers, speakers and attendees). We used the Viewfield module so that conference administrators simply could add a page to a conference and include one of these predefined Views listings. Because the page itself is a part of the conference, the Views listing filters its results for the context of the conference. This approach also allows the listing pages to include custom text before and after the listings, to create an appropriate URL and to pick up the conference theme



NSDI '12 Technical Sessions

All sessions will take place in Regency 2 unless otherwise noted.

Wednesday, April 25, 2012

8:45 a.m.–9:00 a.m. Wednesday

Opening Remarks and Awards Presentations
NSDI '12 Program Co-Chairs: Steven Gribble, *University of Washington;* Dina Katabi, *Massachusetts Institute of Technology*

9:00 a.m.–10:30 a.m. Wednesday

Big Data
CORFU: A Shared Log Design for Flash Clusters
Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, and Ted Wobber, *Microsoft Research Silicon Valley;* Michae[...] *iego;* John D. Davis, *Microsoft Research Silicon Valley*

This content is available to:
Conference attendees

Resili[...]t-Tolerant Abstraction for In-Memory Cluster Computing
Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica, *University of California, Berkeley*

Zaharia PDF

Camdoop: Exploiting In-network Aggregation for Big Data Applications
Paolo Costa, *Microsoft Research Cambridge and Imperial College London;* Austin Donnelly, Antony Rowstron, and Greg O'Shea, *Microsoft Research Cambridge*

Figure 3. A conference microsite has a custom theme, banner, navigation and content, such as sessions and sponsors.

automatically. It also keeps content creators in the familiar content creation interface rather than requiring them to learn another Drupal system.

Some of the conference pages required even more customization, pushing past the limits of what can be automated and programatically displayed. The conference schedule pages, for example, vary in format between conferences. The staff wanted full control over the markup of these individual listings while still benefiting from dynamic content management of each session displayed on the schedule. For these cases, we implemented a token-based HTML system. With this system, an editor creates a content node of type schedule and is presented with a content creation form including a textarea containing a sample HTML template for the schedule. A list of all sessions within the conference and their corresponding token codes also is displayed. Editors can adjust the HTML as they please and paste in these token codes for each session on the schedule. When displayed, the token codes are replaced dynamically with the appropriate session displays.

Multimedia and CDN

For years, USENIX created videos of conference presentations in a variety of formats. In the process of moving these videos to the new site, we scripted their conversion to the Ogg and H.264 encoding. Similar work was performed with existing audio files. All media files now are stored on Rackspace Cloud Files CDN and referenced by Drupal via link fields. When links are pasted into these link fields, they are displayed automatically as HTML5 video available in WebM, H.264 or Ogg, with Flash as a fallback option.

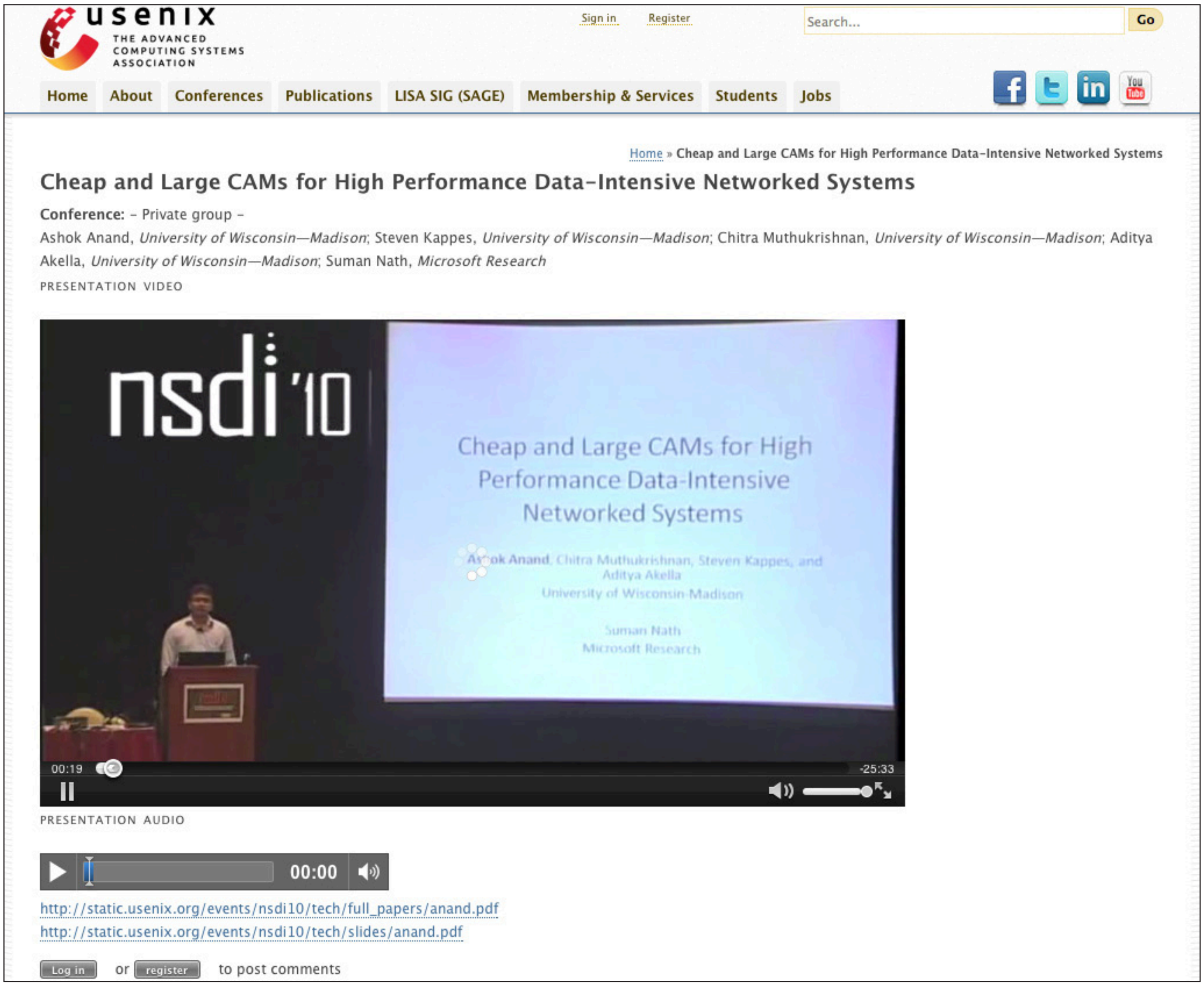
The Rackspace Cloud Files CDN is also used to host public PDF files on the USENIX Web site. We built a custom integration using the PHP Cloud Files API to queue the bidirectional transfer of these files between the Web site and the CDN as dictated by access permissions. As a result, content editors can manage files on the site without concern for the CDN and the appropriate links are shown to site visitors.

Data Migration

The legacy USENIX content was a migration challenge because it was all hand-written HTML. Typically, content migration can make use of an existing database or at least consistency created by a templating system. This content was not only hand-written but it also had been added on-line since 1993, so that there were major changes to the markup over time. We began the migration process by identifying content that was critical to the new USENIX Web site as opposed to that which could be maintained in its legacy form. Of the critical content, we distinguished between content that had enough structure to be scraped programmatically, and that which required manual migration. There was also a WordPress blog that was migrated into Drupal, but WordPress-to-Drupal migration is a trivial process, thanks to WordPress Import and WordPress Migrate Drupal modules.

Legacy Content

Some content was not critical for full integration into the new site. For example, driving directions to a conference in 1995, while fine to archive, was not worth migrating. We used the old site's content directories to create a list of all the URLs on the old site and scraped each page's full HTML into a Drupal node. This legacy content, which included Listserv discussions, was indexed by the Apache Solr search engine so that it can be found



Home » Cheap and Large CAMs for High Performance Data-Intensive Networked Systems

Cheap and Large CAMs for High Performance Data-Intensive Networked Systems

Conference: – Private group –

Ashok Anand, *University of Wisconsin—Madison*; Steven Kappes, *University of Wisconsin—Madison*; Chitra Muthukrishnan, *University of Wisconsin—Madison*; Aditya Akella, *University of Wisconsin—Madison*; Suman Nath, *Microsoft Research*

PRESENTATION VIDEO

nsdi '10
Cheap and Large CAMs for High Performance Data-Intensive Networked Systems
Ashok Anand, Chitra Muthukrishnan, Steven Kappes, and Aditya Akella
University of Wisconsin-Madison
Suman Nath
Microsoft Research
00:19
-25:33

PRESENTATION AUDIO

http://static.usenix.org/events/nsdi10/tech/full_papers/anand.pdf
http://static.usenix.org/events/nsdi10/tech/slides/anand.pdf

Log in or register to post comments

Figure 4. A treasure-trove of conference recordings was made available for all browsers and devices.

in site-wide searches. Viewing these Drupal nodes of legacy content results in a redirect to the page on the real legacy site, now hosted at **http://static.usenix.org**.

Scraped Content

We used the Simple HTML DOM parser to traverse sections of the old USENIX Web site and programmatically create new Drupal nodes with dates, files, images, text and node reference fields. These imports had to be repeated as new content was added to the legacy site, so they were written as Drupal modules containing custom Drush commands. These commands took a URL and scraped its HTML, parsed it into arrays of data, built Drupal objects representing content, files, products or categories and saved these programatically. This approach was used to import some conference content, magazine content and Short Topics in System Administration books content.

Manually Migrated Content

Conference proceedings are arguably the most important content on the site. The proceedings needed to be highly structured to allow connecting content from the same presenters, connecting content from the same conferences, organizing content by awards won and listing content by date. Media on these proceedings including PDFs, slides and videos also needed to be structured. The original content was extensive and lacked a consistent structure that would be needed for scraping.

To facilitate the necessary manual migration of the proceedings of hundreds of conferences, we built a custom migration interface into the development site using the Views Bulk Operations and Workflow modules. We hired contractors to use the interface, which consisted of a Views listing of each legacy conference and its status (incomplete, complete, duplicate or not applicable). Each legacy conference was a link to a data entry form that consisted of an iframe showing the legacy proceedings of the conference and a form for entering in fields, such as title, date, authors, slides and video. We designed the structure of the imported content to make import as quick as possible for the manual process. After entry of thousands of conference proceedings, we converted the resulting nodes into their final data structure: nodes of type session, paper, discussion and speaker.

Design and Theme

The USENIX design was started by Nica Lorber of Chapter Three and completed by our Creative Director, Mason Wendell, in collaboration with USENIX staff. Themes were built with Sass/Compass using the Coding Designer Survival Kit (**http://thecodingdesigner.com/survivalkit**).

Credits

- Drupal development and design by Zivtech: Jody Hamilton (project lead), Howard Tyson, Matt Klein, Steve Heise, Mason Wendell, Sean Wolfe, Benji Davis, Stephen Haslett, Laurence Liss, Justin Randell, Andrew Morton and Meghan Palagyi.
- Design by Chapter Three: Nica Lorber.
- Hosting by Pantheon: Josh Koenig and David Strauss.
- USENIX project leads: Casey Henderson, Jane-Ellen Long and Anne Dickison.
- Video encoding: Joseph Schwartz via USENIX.
- Salesforce development by Heller Consulting: Bran Scott and Kim Kupferman.■

Jody Hamilton is CTO and partner at Zivtech (http://zivtech.com), a Philadelphia-based open-source consultancy. She has been a Drupal developer since 2006 and contributes to core and contributed modules on drupal.org.

Contributing to Drupal: Open for All!

Drupal's contribution process is open and straightforward. It's also the fastest route to mastering this CMF.

KOJO IDRISSA

Drupal is an open-source Content Management Framework (CMF) that allows you to build a Content Management System (CMS) that fits your needs. This article doesn't go into detail about the differences between a CMS and a CMF. The article "Drupal Is Not a CMS" (**http://www.palantir.net/blog/drupal-not-cms**) by Larry Garfield of Palantir (known as "crell" on Twitter and IRC) does an excellent job of explaining this distinction. What's important to readers of this article is the basic structure that makes Drupal a framework. There's a core (analogous to the Linux kernel in many ways) that contains the basic Drupal functionality. This is what you get when you download Drupal. The additional functions come from contributed modules, which add capabilities to Drupal's core.

This article focuses on Drupal's core, how you can contribute to the project and why you'd want to. I'll start with the why, and then move on to the various methods of how.

What Is Drupal and What Makes Up Its "Core"?

Drupal Core is the basic functionality common to any Drupal-based CMS. Like the Linux kernel, it's not Drupal without core. When you download Drupal, you're downloading core. In addition to core, there are several optional pieces of software (called Modules in Drupal-speak) that add useful functions to the core. These range from the nearly critical modules like Views and CCK, used on *most* Drupal projects, to modules meant for very specific use cases.

Why Would I Contribute to Core?

If you already *know* you want to contribute to Drupal core, skip ahead to the "How" section of this article. There are several reasons you would want to contribute to Drupal core, mostly related to how you use Drupal or plan to use it.

As mentioned earlier, Drupal is a CMF, allowing you to build a custom CMS that suits your needs more easily. In some cases, that may mean combining pre-existing contributed modules. If you're someone who's using Drupal to build Web sites or a custom CMS (perhaps an intranet), but you're not planning to make any *changes* to Drupal (such as adding new code to a module), making contributions to core can give you a better understanding of how the system works under the hood. This, in turn, can help you make more educated decisions about which contributed modules to use as you build your project. As an example, you may have a scenario where site speed is critical. There may be two modules that perform a similar function, but one may have been built with an emphasis on ease of use/configuration (which we all like), while the other may have been built for performance. A better understanding of how Drupal core works can help you decide what's best for your use case and what to do when your needs change—and, you know they'll change.

In some cases, there may be certain, specific functionality you need or want that doesn't already exist in Drupal. In that case, you can add functions to an existing module or create a module of your own. Please, don't hack core. Drupal is written in PHP, and core is designed in a modular fashion to accept these new functions. Contributing to core will give you stronger understanding of the best ways to create your new modules to best work with what core or other existing modules already provide. You may not need to create a complex new module—perhaps all that's needed is a small extra bit of functionality. The more you know about what core provides and how it provides it, the less you'll have to create. Remember, as Eric Raymond says, "No problem should ever have to be solved twice" (**http://www.catb.org/esr/faqs/hacker-howto.html#believe2**).

The above examples assume you're someone who's already using Drupal but not making contributions to core. If that's the case, consider this: Drupal is a growing and evolving system. By contributing to core, you're in a great position to learn more about where Drupal is going in the future. As an example, the current version of Drupal is Drupal 7. Drupal 8 is going to be *awesome*! How do I know that? Making small contributions to core keeps me in the loop about what's happening. But, what about people just starting with Drupal?

Say you're a system/network/database admin who's had Drupal "thrust upon you" by someone higher up. *That* never happens, right? I had a sysadmin friend recently tell me that someone in her organization was "excited" about using Drupal. What's that going to mean for my friend? You probably don't care, but you *do* care about what it would mean for you, don't you? Contributing to core can help you get a much better understanding of how a Drupal system may impact the system/network/database resources you administer. This is especially true once you figure out what your organization wants to do with Drupal. I look at specifics regarding that in the "How" section of this article.

What if you're the "Web person" for your organization who's had Drupal "thrust upon you", but with a mandate to make it usable, "pretty" and accesible or to "look like *our* company, not every other Drupal site"? First, realize how lucky you are to have higher-ups who are enlightened enough to choose Drupal. They could have thought Flash was "the next big thing". Second, realize that contributing to core can help you learn the best ways to do that. Again, more details on that in the "How" section.

If you're someone new to Drupal, you'll probably want all the help you can get. The people best positioned to help you *use* Drupal are the people who *built* Drupal. How can you meet these people? By contributing to core.

How Can I Contribute to Core?

The first thing you have to do to before you can make a contribution to Drupal core is register for an account at drupal.org. This allows you to do all sorts of small things immediately, such as update documentation. You can contribute in several different ways. For the *Linux Journal* audience, I'll first focus on the "standard" approach, which is more developer-centric and involves using the Drupal Issue Queue. After that, I'll go into some ways even a "non-coder" or "non-technical" person can contribute.

Code Monkey Climb Ladder, Help Drupal!

The approach most developers (but not *just* developers) will be comfortable with can be seen on the Drupal Ladder (**http://drupalladder.org/ladder/ee503327-50be-1904-8d04-9499098cad64**). The Drupal Ladder is the result of an effort to get more people involved in contributing to Drupal core. It is made up of "lessons and materials to help people learn about and contribute to Drupal" (**http://drupalladder.org/node/1**). It's also worth noting that Drupalize.me, an on-line repository for Drupal training videos created by Lullabot, has a free "Learn the Drupal

Ladder" (**http://drupalize.me/series/learn-drupal-ladder**) series of videos available for viewing. At the time of this writing (August 2012), four videos are available, going through the "Getting Started in the Issue Queue" rung of the ladder, with at least two more in the works. Those are excellent resources as well. The first two steps are basic setup. If you've already got a local Web development environment and Git setup, feel free to skip ahead to step 3.

Step 1: Get Drupal Installed Locally

This involves setting up a local *AMP stack (with the "P" being PHP) and downloading/installing the latest version of Drupal. For most *Linux Journal* readers, this shouldn't be a problem, as most Linux distros already include this software. If you're not running Linux, you've got a couple options. You can use WAMP (**http://www.wampserver.com/en**) for Windows or MAMP (**http://www.mamp.info/en/index.html**) for OS X. Drupalize.me has free videos to help with installing WAMP (**http://drupalize.me/videos/installing-wampserver**) or MAMP (**http://drupalize.me/videos/installing-mamp-web-server**) for those needing help. A second option, for both platforms, is the Acquia Dev Desktop. This is an *AMP package created by Acquia, a Drupal company started by Drupal project founder and lead, Dries Buytaert. Instructions for installing the Acquia Dev Desktop can be found in the second video in the "Learn Drupal Ladder" series mentioned earlier.

After you've got your *AMP stack going, it's just a matter of downloading the latest version of Drupal. If you're happy with the Linux command line, the "Quick install for developers (command line)" (**http://drupal.org/documentation/install/developers**) approach should work just fine, starting with:

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

Otherwise, you can go to **http://drupal.org/download** for the latest version of Drupal 7 (which is the current version of Drupal as of August 2012), or you can go to the releases page (**http://drupal.org/node/3060/release**) to find anything from Drupal 4.7 to the latest update to Drupal 8 (under development now). Refer to Drupal's Installation Guide for further details.

Step 2: Install Git

Drupal uses Git for version control, and if you're going to be helping with the Issue Queue, you'll need it installed locally as well. Instructions for installing Git are beyond the scope of this article, but once you've got it installed, there is a set of instructions specific to working with Drupal core (**http://drupal.org/project/drupal/git-instructions**) available for those new to Git or Drupal. Don't forget, there's also a lesson at the Drupal Ladder site (**http://drupalladder.org/lesson/027b5839-7a74-af04-6905-fee2d01c7ef4**) and a video at Drupalize.me. With the first two setup steps out of the way, you're ready to dive into the Issue Queue and start meeting Drupal Folks.

Steps 3–5: Using the Issue Queue and Interacting with the Drupal Community

Now, I'm going to depart a bit from following the Drupal Ladder, by outlining the basic steps for dealing with a bug in Drupal core, which probably are similar across open-source projects. What's important to note is that you're helping by taking part in *any* of these steps. Please note, this is *my* "broad strokes" overview of the process. It's not an "official" outline or process document. I'm sure I'm leaving out some details, but this will more than suffice for an introduction:

1. Someone has a problem and adds an issue to the Drupal Core Issue Queue (**https://drupal.org/project/issues/drupal?categories=All**). When doing this, be as detailed as possible about the conditions that cause the bug to appear. People will need to be able to reproduce it. Screenshots help. You need to have an account on drupal.org to post an issue.

2. Someone working on the issue queue tests an issue to see if they can reproduce it. If they can't reproduce it, they'll post a comment for that issue, probably asking for more information. This will alert the original poster and anyone else who may be "following" that issue. If they can, they'll add a comment noting this and perhaps move on to the next step.

3. Someone sees an issue they know how to fix. They may explain what needs to be done (this is often done by more experienced developers when they encounter "easier" problems) or create the patch they think will fix the issue. When a patch is created, it's submitted by attaching it to a new comment in the issue queue. Drupal's automated testing system (SimpleTest) will do basic testing on the patch.

4. Someone sees a patch that has passed automated testing and will test it manually on their own local Drupal install. They'll include before-and-after screenshots of the behavior that created the bug and how it's been fixed (or not fixed).

5. Someone with commit access (a small group of people, **https://drupal.org/node/3060/committers**) will commit the successful patch to Drupal core after it's been successfully Reviewed and Tested By the Community (RTBC)

Focus on Your Strengths

My use of the word "someone" in the steps above was important because *you* could be the "someone" in *each* or *any* of those steps (except the last one) in the process, but for a different issue. It all depends on the issue and your abilities. As an example, if you know how to download a file from the Web, take screenshots of what's on your computer screen and upload a picture to the Web, you can do step 4 above. You just need to learn the first two "rungs" of the Drupal Ladder (install Drupal locally and install Git—remember, there are free videos to show you how), and you're ready to go. If you don't know how to create or apply a patch with Git, the Git instructions for working with Drupal core (**http://drupal.org/project/drupal/git-instructions**) tell you how. This step is a critical step, and if that's all you *ever* do to contribute to Drupal core, you'll be appreciated more than you know. But, it also could be a great entry point to other types of contributions.

Similarly, step 3 above provides an opportunity to focus on your areas of expertise. There's more to Drupal than just PHP. There's a lot of markup (**https://drupal.org/project/issues/drupal?categories=bug&component=markup**), including HTML5, CSS (**https://drupal.org/project/issues/drupal?categories=bug&component=CSS**) and JavaScript (**https://drupal.org/project/issues/drupal?categories=bug&component=javascript**) in Drupal core. Each category has issues. The Drupal community is heavy on "back-end" developers, but if you're good with the "front-end" stuff, you may be able to point out, suggest or create patches to core issues quickly, without ever leaving your comfort zone. Again, once you've climbed those first two rungs of the Drupal Ladder, you can go to town.

Strength in Numbers: Core Mentoring Hours

One of the best benefits of working with Drupal is the great community. It's honestly one of the biggest reasons I've spent so much time learning more about Drupal. More than just a CMF, Drupal, like Soylent Green, is made of people—great people. You can leverage those great people as you learn to contribute by participating in Core Mentoring Hours (**http://drupal.org/core-office-hours**).

Core Mentoring Hours (which you may sometimes hear called "Office Hours"—it was originally called Core Office Hours) got started in August 2011 as an effort to get more of the Drupal community involved in helping with contributions to core. As with many projects, you find a small group of people carrying a lot of the weight. If more people could be recruited to help with some of the tasks (like steps 3 and 4 above), that would create two benefits. First, it would leave the Drupal architects free to focus on higher-level stuff (instead of fixing typos in text strings, an actual issue I've worked on). Second, it would help "build" *new* Drupal architects by letting members of the community learn more about the inner workings of Drupal. The benefit for new contributors is that there's now an official, well-formed (and constantly evolving) "on-ramp" for new contributors.

Currently, Core Mentoring Hours happen twice each week, Tuesday, 02:00–04:00 UTC, and Wednesday, 16:00–18:00 UTC. Here in Texas, that's 9–11pm Mondays and 11am–1pm Wednesdays. The Core Mentoring page has time-zone converters so you can find your local times. If you can make yourself available during one of those time slots, go into the #drupal IRC channel on freenode.net and announce that you're there for Core Mentoring Hours (**http://drupal.org/irc**). You promptly will be put to work and/or assisted with questions you have about how to contribute. I participated in the *first* COH/CMH last August, and I've learned quite a bit.

Step 6: Pick Your Spots, Call Your Shots

Remember those examples I mentioned about the people who had Drupal "thrust upon them"? Remember when I said you'd be able to find ways to contribute that would benefit your specific functions? This is where the various Drupal core subsystems come into play. My sysadmin friend may be more concerned with the cache (**https://drupal.org/project/issues/drupal?categories=bug&component=cache+system**) or database (**https://drupal.org/project/issues/drupal?categories=bug&component=database+system**) systems and how they work. Looking at bugs related to those systems can help expand her understanding of how they function. If her organization is using MySQL (**https://drupal.org/project/issues/drupal?categories=bug&component=mysql+database**) or PostgreSQL (**https://drupal.org/project/issues/drupal?categories=bug&component=postgresql+database**), she may want to focus more on those. The "Web person" may want to focus more on things like the markup, CSS or JavaScript subsystems I mentioned earlier. Or, if the site has a lot of community activity, maybe looking at the Comment module (**https://drupal.org/project/issues/drupal?categories=bug&component=comment.module**) would be helpful.

If you know what parts of Drupal you're more interested in, point that out while you're in Core Mentoring Hours, and you can work on those areas. Or, work on it on your own. Another option is to look at the various Drupal 8 Core Initiatives (**http://drupal.org/community-initiatives/drupal-core**), which focus almost exclusively on specific areas, such as configuration management, Web services or mobile. Discussion of those topics deserves its own article, but take a look at the Initiatives page and see what looks good.

I'm No Code Monkey Genius, So I Can't Contribute

There are a number of ways non-coders can contribute. *Every* issue in the core queue *should* have a clear, concise summary. They don't. The Issue Summary Initiative (**http://groups.drupal.org/node/167534**) looks to fix this. If you can read and write English, you can help here. Change notices (**http://drupal.org/node/1314040**), although slightly more technical, involve documenting changes that already have been made. So, if there's a patch that changes an API or adds/removes a function, a Change Notice is needed. Here's an example of one I worked on for a new function that themes (styles) date/time information (**http://drupal.org/node/1441334**). I didn't write the patch, just the Change Notice. Finally, as a member of drupal.org, you can update most of the documentation on the site, and there is a lot! As an example, while I was writing the paragraph on Core Mentoring Hours, I updated the time-zone conversion link for 02:00 UTC Tuesday. It was going to 02:00 UTC Monday, probably because the Mentoring Hours *started* on Monday/Wednesday. It was a simple fix, but each bit helps.

I've done all three of the things mentioned above. If you're curious about any of them, you can take a look at the docs I've linked to or ask in Core Mentoring Hours.

Conclusion

Drupal is an excellent CMF that offers a lot of functionality out of the box, as well as for people wanting to build something custom. In a more general sense, Drupal is also an open-source project with a very welcoming and supportive community. In the words of Angela "webchick" Byron, open-source evangelist, a Drupal core co-maintiner and April 2011 *Linux Journal* cover girl (**http://www.linuxjournal.com/content/interview-angie-byron-drupal**), "Drupal is a Do-ocracy". It's not about who you are; it's about what you do. My own experiences have confirmed this. There's very little cliquishness, and anyone who's willing to learn about the community standards is welcome to contribute. This article exists only because I've come across so many smart and helpful people contributing to the Drupal community. Join us! ■

Kojo Idrissa is not an MBA. He has an MBA. There's an important difference. A holistic mixture of tech guy, business guy, China guy and educator, he fits into more than one space. When he's not being "that guy", he's learning new ways to build things and solve problems with technology, improving his Chinese or honing his teaching skills. He's co-SIG Leader (and resident "Drupal Guy") of the HAL-PC Web Design SIG and a member of the Houston Drupal User Group. He's "Transition" in the Drupal IRC channels and on drupal.org.