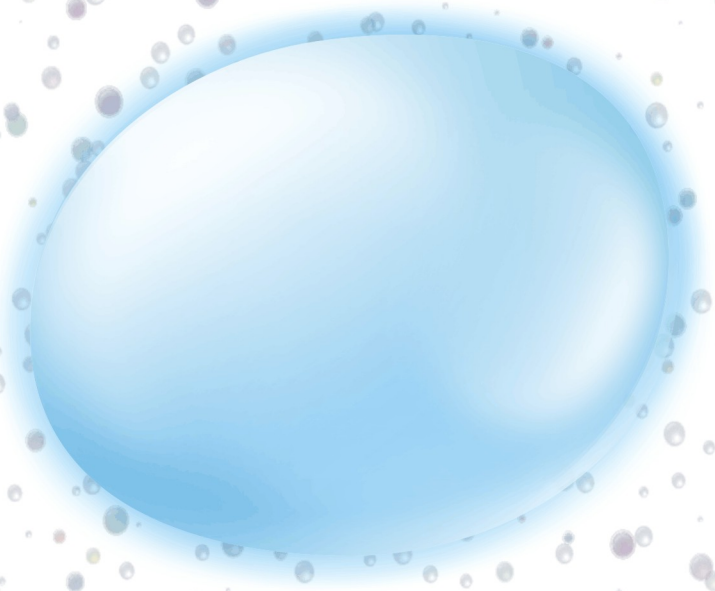
DurableDrupal
Lean



Tool up for solid web apps with a powerful Drupal distro

by

Victor Kane

DurableDrupal Lean

Tool up for solid web apps now and maintain them for years with a powerful easy to install Drupal distro

Victor Kane

Contents

Introduction

Tool up for solid web apps now and maintain them for years with a powerful Drupal distro

This book is for web app development teams, called upon by their clients to move their business model to the internet, their back-office to the cloud.

This book will empower you and your team, no matter what your level of expertise, to simplify the building of solid solutions for your clients' real needs, and to facilitate in a productive and cost-effective manner the maintenance these solutions will be crying out for from the moment they are born.

In the process, you will actually be evolving for yourselves a lean web factory to output durable solutions that can be maintained for years to come.

Change, maintenance, continous delivery and the long haul

Successful web developers and agencies know that the real challenge is how to embrace continuous innovation and delivery throughout the web app life cycle.

Delivering websites and web apps is only the beginning. The real question for success in web development is how to continue to ply your client with continual fixes and releases in order to fix problems and satisfy change requests, to optimize value at all times.

Because constant change is inevitable, and ever-present throughout the web app life-cycle:

- First in the minimum of 40% in changes that will be requested in the weeks or months of development.
- Secondly, in the feature and usability requests the solution will have to assimilate in order to maintain its value over time.
- Thirdly in the security threats that will have to be dealt with in as fully an automated way as possible.
- Fourthly in the architectural and deployment changes that will be required as time goes by, in the struggle to scale with growth and cut costs with new technological breakthroughs.

Because it's not about the "single click site generation" and easy scaffolding any decent framework or "hosting" solution offers. We'll soon see how to fire up a framework in seconds, in just a few clicks, but the devil is in how we empower ourselves with a process that is going to facilitate, stabilize and simplify our work in the face of this constant change. By using the most exciting best of breed industry standard tools, like Ansible, Vagrant and Docker, Cloud-based hosting, and the all important Git branch model; just to mention a few of the things we will be working with here.

The theme of embracing change, while not new at all, marks the difference between success and failure, is born of the inevitable failures as well as the successes we all live through as we gain experience, and runs throughout the book so that web development can be fun instead of being a constant source of stress, code and maintenance debt.

We're here for the long haul.

And we're using Drupal[1] now.

And we're using Lean process now.

Overview of Book Contents

So here's what we'll be doing:

- Chapter 1: First Lean DurableDrupalDistro Project
  - First meeting with the client
- Chapter 2: Setting up just enough shop to deliver our first web app
  - Team consciousness: client reps and dev team together
  - Shop infrastructre
    - Central code repos for all deliverables using the Git branching model
    - VPS MVP instances
      - Testing & Integration instance
      - Staging instance
      - Production ready instance
    - Every team member has their own dev environment automatically provisioned on their own laptop
    - Tools for running our agile developoment process
  - Fork the distro and make it our own
  - User stories and doing it with agile
  - Delivery as client empowerment
  - Retrospective: We did it! But…
    - Agile is the new waterfall (pixel perfect design -> "implementation")
    - Team (devs and designers, devs and client reps) couldn't really work in parallel
      - The tangle of User Stories and Design

[1] https://www.drupal.org/

        * Client took delivery of a monolithic blob he wasn't sure how to use
            · We had to do special training (not to mention train the trainers)
            · Client reps began a slew of change requests that could have come earlier if they had been familiar with the system
- Chapter 2: Pain driven development
    - New project!
    - Value hypotheses
    - Design studio for nerds… and for the client too
    - Getting the team working in parallel
- Chapter 3: MVP prototypes
- Chapter 4: Grabbing the legacy content This should be postponed for migrations book
- Chapter 5: Candidate architecture
    - Choices panic: see Appendex 1 xxxxxx
    - We're evolving on top of best of breed
- Chapter 6: Build the recursive MVP of MVP's
    - Every user story is an MVP
    - No project is ever anything more than an MVP
- Chapter 7: Go team test team go go go The advantage of using frameworks isn't just that they're tried and tested, but that they empower testing and TTD
- Chapter 8: Build, deploy, maintain
    - Rinse and repeat over "development"
- Appendix 1: Overcoming Choice Panic - What frameworks should we use?

  We need to free ourselves from choice panic which can paralyze us or disrupt our work. Dialog is necessary to ensure that the project stakeholders (all of them) are on board with the major framework optionss. We need to have this discussion sooner rather than later in order to share a common perspective and be ready to get down to work. In order to achieve this, we will be facing some tough questions together:
    - Building from scratch or not? Frameworks or not?
    - Proprietary or not? Open-source or not?
    - WordPress or not?
    - Drupal or not?
    - Drupal 8 or not?
    - Backdrop or not?
    - Vanilla Drupal 7 or not? Distro or not?
    - Presenting DurableDrupalDistro: fork it, tailor it, maintain it, branch and release it over all your projects!
    - Preparing to migrate gradually all along, so we'll be ready when the time comes
- Appendix 2: Team Workflow and Provisioning

Formatting Conventions

This book follows industry standard formatting and typographical conventions[2] for web development books.

Obtaining and using the DurableDrupalDistro and other code examples

This is a book we need to work through together, so a starter Drupal Distro has been provided.

> For team dev, integration, staging and production DurableDrupalDistro provisioning, fork and adapt (or directly download and use) the Ansible playbook for setting up Durable Drupal Distro[3].
>
> Or grab the DurableDrupalDistro[4] itself and build out as you like.

At a minimum

Thanks to all the modern tools that have come along, all you really need is a laptop with some disk space and a decent amount of RAM. And we promise not to dirty it up with a lot of "server" stuff, thanks to Vagrant, Virtual Box, Ansible and Docker. We'll be setting up complete self-contained server replicas that won't affect your regular configuration at all. And you'll be able to follow through all the examples throughout the book.

Contact and feedback

Victor Kaneâ€¦ Book page… mailing lists

Acknowledgements

But more than a book this is a community of teams. Dive in!

[2] https://leanpub.com/help/manual
[3] https://github.com/victorkane/ansible-vagrant-durable-drupal-distro
[4] https://github.com/victorkane/durable-drupal-distro

Chapter 1. First Lean DurableDrupalDistro Project

To find out what DurableDrupal Lean is all about, let's dive right in and work through a project together using the DurableDrupalDistro. In this chapter we'll show how the team from AWebFactory work together in full parallel cross-collaboration from start to finish. We will then drill down in later chapters with "just enough" detail to see how stuff gets done in this real life project. As always there will be some room for improvement, and this will be caught in the retrospective sessions we have from time to time and at the end of the book.

At AWebFactory any member of the team can start a project by reaching an agreement with a client, form a team and do a job of work.

Project start

Lisa, a team member at AWebFactory, gets a mail from Stuart. He is a very well-known writer of Flash Fiction[5], and has been running a remote workshop for flash fiction writers for decades using a moderated email list. The workshop also has a website, where bulletins about writing opportunities are put up, along with a flash fiction zine publishing selected material written by the workshop members themselves as well as by flash fiction masters. In the mail Stuart tells Lisa they need to redo their website and asks to have a Skype chat about this.

As far as Lisa is concerned, the project has already begun. It may have a very short life if no agreement is made with the client, but it has already been born.

Check out the Project Inception overview for more detail.

So Lisa opens up a project Kanban board using Trello with the title FlashFictionWorkshop. She adds the typical three columns to start out with (To Do, Doing, Done), she adds a first card, **MVP Vision**, to the To Do list in order to structure that first meeting. She also adds a **Read Pamelyn Casto's Flashes On The Meridian: Dazzled by Flash Fiction**[6] card for extra credit domain knowledge homework.

She populates the **MVP Vision** card from the MVP Vision Template[7], reads Pam Casto's article and drags that card to the Done column.

[5] http://en.wikipedia.org/wiki/Flash_fiction

[6] http://www.heelstone.com/meridian/meansarticle1.html

[7]

First meeting with the client

He told me they've had a webmaster who managed the content on their site for them, but he seems to have disappeared and they've lost access to their old stuff. Actually they've been wanting to re-do their site for some time, so they have some ideas, but now they're stuck. They also want to be able to manage their content without depending on anyone else for that.

Stakeholder detection. Existing solution survey. I hit him with Problem, Market, Product just to get initial feedback, and explained about us needing to postpone jumping in and talking right away about design and functionality ideas in favor of first clearly understanding current pain points and problems, and understanding what the organization's valued outcomes might be. And that afterwards we would all brainstorm about designs and functionality that could be tested out to see if they might achieve those objectives. As usual it was pretty hard to sell this point since clients always want to talk about widgets, colors and what blocks should be handy to click on right away, but I think we were all on the same page, and I came away from the meeting with some good notes. The good news was that Stewart agreed to be readily available for ongoing feedback and understood about our methods of continuous delivery. TODO: What the client can understand in a first meeting is what fits into this chapter. So, simple explanations over buzz words.

Durable Drupal Lean in a Nutshell

You can get a nice clear overview of the process, if you prefer, in Appendix 1. DurableDrupal Lean Process Overview, which serves also as a handy future reference, as well as in the links in the rest of this chapter. Feel free to skip those links, though, if you prefer the "practical" overview presented here first.

It might be a good idea to read this chapter twice, once to get a quick feeling for how the process is used, and a second time to dig a little deeper!

TODO The Overview and Templates and all instances of their use can be found in the TODO WTF! node, files … SOCORRO … live style guide … live user hypotheses …

Forming the team

It was a minimum team but had to have the following areas covered (with some team members covering more than one role):

- Product owner
- Process coach
- Software Architect
- User experience/User interface designer
- Graphic designer

- Multimedia designer
- Structured Content specialist/Information architect
- Front-end developer
- Back-end developer
- Marketing specialist
- Content creator/Copywriter
- DevOps specialist

Notice the absence of Project Manager. It's important to escape the rockstar fantasy and indispensable hero syndrome. It's the team the whole team and nothing but the team. It just works.

The exact definition of each of the roles will become apparent by the end of this chapter and throughout the book. If you need an abstract definition, see Appendix 2.

Single Theme project encompassing 3 Sprints aimed at MVP

I planned for a Lean process with the team, blending Lean with Scrum with the Theme approach discussed in Gothelf which avoids output handoff, deliverables and waterfall in favor of everyone working in cross-collaboration based on … the whole team working on the same thing at the same time (TODO: redact). A single set of 3 2-week sprints was going to be enough to deliver and test an MVP after initial prototypes and a first MVP had been tested and corrected in the first two sprints.

Sprint 1

Kick-off Assumptions Declaration and Design Studio meeting

- Problem statements
- Assumption declarations
- Market, Problem and Product preliminary hypotheses
- 2-day break for GOOB and Problem & Market Validation

GOOB and Validation

Main Assumptions Declaration and Design Studio meeting with feedback

- Complete hypotheses cycle
- Design studio
- Framework discussion (see Appendix 1)

Iteration Planning Meetings

- User Stories
- Backlog generation for Sprint
- Planning for interspersed usability and prototype/MVP product value testing including two cross-collaboration meetings

Sprint 1

- IPM as above, and
  - Selection of DurableDrupalDistro based on discussion in Appendix 1
  - Provisioning and Team workflow presentation (see Appendix 2)
- Provisioning for office, server, individual team members
- Workflow for jobs of work integration and continuous build
- Backlog
- Testing
- Conversations
- Guaranteeing creative time for all
- Prototype delivered and tested

Sprint 2

- IPM as above
- Backlog
- Testing
- Conversations

Sprint 3

- IPM as above
- Backlog
- Testing
- Conversations
- Delivery
- Testing
- Retrospective meeting

TODO work in artifacts (3rd appendix?) plus tools from both books (Klein's 9 tools, etc.)

TODO just do it

—– TODO rework the following on the basis of our actual Lean process followed for the chapter project.

Exactly how we build this thing

- Getting the team together
  - Provisioning
  - See Appendix 1 xxxxx on why the team chose DurableDrupalDistro as the framework of choice for this project.
- First planning session
  - Value hypotheses
  - Testing
- Second planning session
  - MVP Specification based on “what we now know”
  - Establish initial sprint backlog
- Provisioning for development and staging
  - Tailor distro to project
  - Provision team with local VMs on their laptops
- First user story
  - Design studio
  - Estimates
  - Acceptance test
- Successive user stories
- Sprint Acceptance
- Retrospective

Getting the team together

Kickoff session

See https://github.com/victorkane/lit-drupal-lean/issues/3 and refactory this whole outline accordingly

Value hypotheses

Testing

Second planning session

MVP Specification based on “what we now know”

Establish initial sprint backlog

Provisioning for development and staging

Tailor distro to project

Provision team with local VMs on their laptops

First user story

Design studio

Estimates

Acceptance test

Successive user stories

Sprint Acceptance

Retrospective (or Agile is the new Waterfall)

Dig a little deeper

Now that you have seen a live example, now would be a great time to learn more about the DurableDrupal Lean process in Appendix 1. DurableDrupal Lean Process Overview which also references all Guides and Templates. You can also take a look at the Guides in Appendix 1A. DurableDrupal Lean Process. Guides and at the Templates in Appendix 1B. DurableDrupal Lean Process. Templates

So in this chapter we ran with the DurableDrupalDistro, built and delivered our project and installed in on our maintenance stack. After which we complied with the solemn retrospective and reviewed our list of lessons learned.

For the rest of the book, we are going to drill down into a more detailed view of the building, delivery and ongoing validation (the new maintenance) of the Literary Workshop and how the current maturity and richness of the Drupal 7 distro eco-system used in conjunction with an evolving distro and process can streamline our work. So on to Chapter 2, where we'll get to experience the Kickoff session up close.

Chapter 2. Value Hypotheses for the project

Chapter 3. Functional prototypes

Chapter 4. Grabbing the legacy content

This should be postponed for migrations book

Chapter 5. Candidate architecture

Chapter 6. Build the recursive MVP of MVPs

Chapter 7. Go team test team go go go

> If I hear “Get out of the building” one more time I think I’ll scream

It’s not just about code that runs right, even though it must.

It’s about value! People’s needs!

Chapter 8. Build, deploy, rinse, repeat

> If I hear “Rinse and repeat” one more time I think I’ll scream

“Whatever happened to getting it right the first time?” muttered Jason, sick and tired of stuff he thought was already done being “postponed” to “the next iteration”.

Appendix 1. DurableDrupal Lean Process. Overview

What follows is an overview of how to make use of the process. There are also links drilling down to templates you can use immediately as your project gets going.

If you would like more background as to why Agile and why Lean, please feel free to skip first to the Agile Software Development. Background and Resources section.

We will describe the process via a listing of typical chronological steps and activities, although it is important to bear in mind that certain projects will want to tailor the use of the tools and artifacts in accordance with their own specific characteristics.

TODO place this snippet somewher: For example, just in order to complete the initial MVP vision, some projects will warrant a fully-functional MVP prototype or test landing page before going any further, while others will base their first prototype on a set of user stories.

Project inception

Initial contact with the project owner

Somewhere, somehow the project is born. A telephone call or an email is received, a comment made during a chance encounter on the street or at the mall. The job of the agency member is to initiate an organized process involving a job of work. As such, an invitation is issued for a meeting, either in person or through a remote service such as Skype[8] or Google Hangouts[9]. Once the invitation is accepted, the project is born.

At this point, an agency member opens up a Kanban board[10] for the project. DurableDrupal Lean works very well with the light-weight and powerful Trello[11], but you can use any other alternative you are familiar with, including the following (most have some kind of free account):

- Build your own
  - Make a personal Kanban board[12]

[8] http://www.skype.com/en/

[9] https://plus.google.com/hangouts

[10] http://en.wikipedia.org/wiki/Kanban_board

[11] https://trello.com/

[12] http://www.instructables.com/id/Kanban-Organization-thru-post-it-notes/

    - You can use a whiteboard with post-it notes and then take photos to capture state
    - Kanban examples[13]
- Mingle[14]
- Pivotal Tracker[15]
- Fulcrum[16] Open source clone of Pivotal Tracker
- Gitban[17] Worth trying since it uses Github Issues directly from your repo
- HuBoard[18] Another Kanban board built on top of Github Issues
- LeanKit[19]

After creating a new board, generally three columns are created:

Whether it's during a meeting with a prospective client or an interview after having submitted an RFP (Request for Proposal), a dialog is started concerning the building of a product capable of solving a specific problem felt by a significant group of people.

Laura Klein, author of Lean UX - Applying Lean principles to Improve User Experience by Jeff Gothelf and Josh Seiden[20] ... m p p

[13] http://leankit.com/kanban/kanban-examples/
[14] http://www.thoughtworks.com/products/mingle-agile-project-management
[15] http://www.pivotaltracker.com/
[16] https://github.com/fulcrum-agile/fulcrum
[17] http://www.gitban.com/
[18] https://huboard.com
[19] http://leankit.com/
[20] http://www.jeffgothelf.com/blog/lean-ux-book/#sthash.22gOyp12.dpbs

Defining Market, Problem and Product

"A market is the group of people you think might want to buy your product…. A problem is the reason that those people are going to use your product… A product is simply the way that you're going to solve the user's problem. It's the end result of what you're building. It's the thing that people, presumably in the target market, are going to pay you money for." (Klein, Chapter 1)

On Validating the Problem

"You are going to discover a problem that exists within your target market that you are capable of solving. Remember, if there's no problem, then there is no compelling reason for people to purchase your product…. You'll know that you've validated a problem when you start to hear particular groups of people complaining about something specific."

On Validating the Market

"Validating the Market: You'll know that you've successfully validated your market when you can accurately predict that a particular type of person will have a specific problem and that the problem will be severe enough that that person is interested in purchasing a solution."

On Validating the Product

"Just because you have discovered a real problem and have a group of people willing to pay you to solve their problem, that doesn't necessarily mean that your product is the right solution…. You'll know that you've validated your product when a large percentage of your target market offers to pay you money to solve their problem."

So it's important to bring up the method of Validation Driven Design right from the start with the client, and even in the first interviews, learn as much as is possible without further testing and research about the Problem and Market the proposed Product is related to. And to emphasize how all the development, even (especially) after delivery will be concerned with validating all three cornerstones.

The DurableDrupal Lean process helps that along by providing an optional MVP Template. There is a guide on how to work with it. It's optional because we never want to impose an obligatory paper-heavy process on anyone. If you feel this template (and the other two or three[21] we include) helps to organize, so much the better. Feel free.

21

Agile Software Development. Background and Resources

Introduction

The difficulty of getting a birds' eye view of any software development process is that it is at its heart a grammar and specialized vocabulary, a set of conventions, tools and structures used dynamically and creatively in real world contexts. When it is used of course, it appears as a logical sequence. Like title, author, sections, sentences, phrases, words and expressions in a blog entry, or a series of instructions in a manual. But any attempt at explaining the grammar runs the risk of reducing the whole dialog to boring and lifeless abstractions.

When we're learning, of course, we need a list of conventions, a vocabulary, and a grammar allowing us to see the options we have in making use of process, and in order to understand the language when we are new to a team.

Remarkably, incremental and iterative software development is nothing new at all, and goes at least as far back as the Waterfall model[22].

Why use process at all

Work is a social process, ever since we gained an opposable thumb. It's also an historical process. Why do we want to engineer things using today's tools and processes? Simply in order to avoid making the same expensive mistakes over and over again, and to avoid inventing the wheel when we tool up for a product.

Waterfall

The baseline in software development is what is called the Waterfall Model[23].

Winston W. Royce's original article Managing the Development of Large Software Systems[24] (interesting that even it includes quite a bit on iteration)

> The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.
>
> The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly,

[22] http://en.wikipedia.org/wiki/Agile_software_development#History

[23] http://en.wikipedia.org/wiki/Waterfall_model

[24] http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

> if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.[1]
>
> Wikipedia: Waterfall Model[a]
>
> [a] http://en.wikipedia.org/wiki/Waterfall_model

In following the Waterfall Model, the various disciplines are executed in a single, non-iterative cycle: Requirements, Design, Construction, Integration, Testing, Installation and Maintenance.

Waterfall may still be the mest process model for your project if it is static in requirements and it is unlikely or prohibitively expensive for changes to be made and there is an insurmountable need for pinpoint estimates in scheduling and resources.

For many other kinds of projects, however, including most website and web app projects, Waterfall can be a recipe for disaster.

What's wrong with Waterfall?

In projects where changes are very likely to occur along the way, or in which technology is changing rapidly, the Waterfall Model accentuates the risk of failure:

- Because of the long delay between initial assumptions and requirements gathering and delivery, the probability of erroneous feature sets being implemented that do not really comply with real user and organizational requirements is greatly amplified.
- Because changes (scope creep) are bound to happen (40% of the requirements is a common minimum figure) but are not recognized officially, either the real costs exceed budgeted costs by far, or else long and even critical delays are often occasioned.
- Since testing is not done on the unit, integration and quality assurance levels, the cost and impact of repairing anomalies is much higher and the overall quality of the product suffers.
- Elevated cost of failure means there is no room for error, and leads to a costly fear of experimentation on all fronts.

Better ways of developing software. The Agile Manifesto

The Agile Manifesto[25], defending a leaner, iterative and incremental approach to software development, bases itself on following 12 principles

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

[25] http://agilemanifesto.org/

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity–the art of maximizing the amount of work not done–is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

This author is proud to have signed the Agile Manifesto in August 2006

Victor Kane: (AWebFactory) Building software must be built around real human needs, and neither the product nor the process shoved down people's throats. The need for software and the need to produce that software must be framed in a process (the process mirrored in the product - Hudson), is optimized by open, flowing structures reflecting those very real needs and their most excellent solutions. Obstacles imposed by greed and the need to defend privilege must be removed. Working people who produce and use software must together manage that process themselves, worldwide.

Scrum

TODO Scrum section

Kanban

TODO Kanban section

Kanban is where Agile and one of its industrial manufacturing predecessars, JIT (Just in Time). And is what gives origin to the today's everyday agile expression "Just enough, just in time".

Bibliography

Agile/Lean

- The Agile Manifesto[26]
- Extreme Programming: A gentle introduction[27]
- Agile Software Development: A gentle introduction[28]
- The Lean Startup by Eric Ries[29]
- Mike Cohn's Scrum and Agile site[30]
- Agile @ Adobe Agile Resources[31]. Excellent comprehensive list of all kinds of Agile and Scrum Resources
- Extreme Programming Explained: Embrace Change by Ken Beck[32]
- Agile Software Development with Scrum by Ken Schwaber & Mike Beedle[33]. Original Scrum book.
- User Stories Applied by Mike Cohn[34]

Lean UX

- Lean UX - Applying Lean principles to Improve User Experience by Jeff Gothelf and Josh Seiden[35]
- UX for Lean Startups: Faster, Smarter User Experience Research and Design by Laura Klein[36]

[26] http://agilemanifesto.org
[27] http://www.extremeprogramming.org/
[28] http://www.agile-process.org/
[29] http://theleanstartup.com/book
[30] http://www.mountaingoatsoftware.com/
[31] http://blogs.adobe.com/agile/2014/12/05/agile-resources/
[32] http://www.amazon.com/Extreme-Programming-Explained-Embrace-Change/dp/0321278658/ref=sr_1_1
[33] http://www.amazon.com/dp/0130676349
[34] http://www.amazon.com/User-Stories-Applied-Software-Development/dp/0321205685
[35] http://www.jeffgothelf.com/blog/lean-ux-book/#sthash.22gOyp12.dpbs
[36] http://www.amazon.com/UX-Lean-Startups-Experience-Research/dp/1449334911

Kanban

- Mattias Skarin's Kanban resource page[37]
  - 10 kanban boards and their context (Examples[38]
- Leankit's What is Kanban article[39]
  - Leankit's Manage Projects with Kanban[40]
  - Leankit's 10 kanban boards examples implementation[41]
  - LeanKanban University - Definition of Kanban Method[42]
- Kanban and Scrum by Henrik Kniberg and Mattias Skarin[43]

Content Strategy

Content Audits and Inventories

- Misty Weaver and The Art of Content Audits Audits, inventories, and the fabulous world of content planning[44]
  - Misty Weaver, on the web[45]
- Doing Content Inventories by Jeffrey Veen[46]
- Drupal Content Audit Module[47]
- [http://growingventuresolutions.com/blog/content-inventory-and-content-audit-views.html

Content Strategy for the Web, by Kristina Halvorson, particularly Chapters 4 and 12

A Checklist for Content Work by Erin Kissane[48]

The Elements of Content Strategy by Erin Kissane

Information Architecture for the World Wide Web by Peter Morville and Louis Rosenfeld

A Project Guide to UX Design by Russ Unger and Carolyn Chandler

[37] https://www.crisp.se/gratis-material-och-guider/kanban
[38] http://blog.crisp.se/2010/12/03/mattiasskarin/1291361993216
[39] http://leankit.com/kanban/what-is-kanban/
[40] http://leankit.com/kanban/online-kanban-board/
[41] http://leankit.com/blog/2010/12/10-kanban-boards-leankit-kanban-style/
[42] http://www.leankanban.com/node/10
[43] http://www.infoq.com/minibooks/kanban-scrum-minibook
[44] https://www.lullabot.com/blog/podcasts/insert-content-here/22-misty-weaver-and-art-content-audits
[45] http://www.meaningandmeasure.com/
[46] http://www.adaptivepath.com/ideas/doing-content-inventory/
[47] https://www.drupal.org/project/content_audit
[48] http://alistapart.com/article/a-checklist-for-content-work

Structured Content Karen McGrane http://wimleers.com/article/drupal-8-structured-content-authoring-experience Drupalcon Portland 2013 Keynote “Thriving in a world of change” http://karenmcgrane.com/2013/05/23/d keynote-video-and-talk-notes/ A separate mobile website: no forking way! http://www.netmagazine.com/opinions/s mobile-website-no-forking-way Jeff Eaton Drupalcon Portland 2013 “building for the post-mobile world” video: https://portland2013.drupal.org/node/1353 slides: https://speakerdeck.com/eaton/building-for-the-post-mobile-world Larry Garfield Drupalcon Portland 2013 “Is Drupal a CMS? Do we even know what that means?” video: https://portland2013.drupal.org/node/3908 slides: http://www.garfieldtech.com/pres is-drupal-cms/#/ Wim Leers Drupal 8: best authoring experience for structured content? http://wimleers.com/article/ 8-structured-content-authoring-experience

A Smallcore Manifesto Adrian Rossouw Where do we go from here? Drupal is not a CMS Larry Garfield COPE: Create Once, Publish Everywhere Daniel Jacobson Content Strategy for Mobile Karen McGrane Adaptive Content Management Mark Boulton Inline Editing and the cost of leaky abstractions Jeff Eaton Content Geography The Art and Practice of Content Assembly Deane Barker

Core Model

- Original 2007 presentation by Are Halland: Core+Paths: A Design Framework for Findability, Prioritization and Value[49] (Also see additional recommended slides on Slideshare)
- The Core Model: Designing Inside Out for Better Results by Ida Aalen[50]
  - The Core Model: Links and Resources[51]

[49] http://www.slideshare.net/aregh/corepaths-a-design-framework-for-findability-prioritization-and-value
[50] http://alistapart.com/article/the-core-model-designing-inside-out-for-better-results
[51] http://alistapart.com/blog/post/the-core-model-links-and-resources

Appendix 1A. DurableDrupal Lean Process. Guides

In DurableDrupal Lean, MVPs scale by being recursive. A bug report or simple change request on up to complicated distributed web applications are MVPs composed of one or more User Story MVPs.

Guide: MVP Vision

Kickoff

-kickoff steps

MVP vision

The MVP vision should be shared at Kickoff.

Appendix 1B. DurableDrupal Lean Process. Templates

MVP Vision Template

Client info

Appendix 2. Overcoming Choice Panic - What frameworks should we use?

If we take the neighborhood youth program website from Chapter 1, or the online community of writers used in successive chapters, the question inevitably arises: what frameworks should we use?

In this appendix we'll be putting Choice Panic behind us and making a firm decision on the CMS framework we'll be using for our content-centric projects. First we'll place all our doubts right on the table for all to see (should we be using Drupal for this project? really?), sift through the options, and choose the best one as our initial hypothesis. In making that decision we'll base ourselves on our current backlog of projects and on the information we actually have to go on right now. We'll explore that particular choice and the characteristics that make it the most versatile and dynamic option open to us. We want to be comfortable with the decision, and have the whole team all on the same page.

Now, also, above and beyond the framework, we are adopting a process which bases itself brutally on what we need and what our clients need, and which abandons vested vendor lock-in of any kind. We can't be married to any one set of frameworks any more. So the framework we select must be able to fit into our adopted, evolving and tailored process at a time when the industry is exploding with change and opportunities.

At the same time, we bear in mind that this is not a one-off question, but one which must be addressed periodically. The thing is, it must be addressed in order to allow you to work in a context of confidence that you are using the best tools for the job, instead of always having that question lurking in the back of your mind.

"Building from Scratch" or not? Frameworks or not?

By starting from scratch what is usually meant is not actually starting with a blank page, but rather the use of light-weight frameworks and libraries over larger CMS frameworks. Light-weight frameworks might give you more freedom and flexibility and may be easier to start using, whereas larger frameworks offer a lot of built-in off-the-shelf functionality together with a large, easy to extend eco-system of plug-ins and add-ons. And using them often constitutes being part of a relatively large community of builders and users, offering possibilities for helping and being helped as well as for networking of many kinds.

Starting from Scratch: Pros

Many developers are attracted towards smaller, light-weight libraries and components because they feel they will be dealing with less complexity, and will be able to be more creative and to "break

new ground". They also feel they will not be stuck with a monolithic system and faced with the task of "turning off" a great deal of the built-in functionality not required for many projects, which is something not always easy to do.

- You are building on top of shiny new technology
- Experimentation with new technologies and frameworks is not only fun but necessary in order to know what the options are.
- When a project is all about things that have never been done before, a completely straight-up ground-breaking disruptive Blue Ocean[52] effort, it only makes sense to find a best-fit candidate architecture which doesn't complicate things with its extra baggage and lowest common denominator features which won't bring anything to the table or may even be a nuisance.

Starting from Scratch: Cons

- If your are developing a content-centric project with many use cases that coincide with the larger CMS frameworks, then it really does not make any sense to reinvent the wheel. Countless features will have to be re-created: authentication and authorization (think: access rights for users and groups), forms management, database management, separation of concerns between business logic, data and the presentation layer, etc. It's a long list.
- The larger CMS frameworks are constantly monitored for security threats and constantly offer security updates when threats emerge. So much so that to some (superficial) eyes the sheer number of security fixes seems to constitute evidence of their being more likely to come under attack. But experience shows this nothing could be further from the truth. One-off apps will need to dedicate considerable time and resources towards security issues on many levels.
- Starting from scratch may need a larger team, with more specialists compared to using a CMS framework.
- It is futile, for many use cases, to attempt to replace the work of whole communities across several years with a small team.
- You will have a harder time finding replacement team than you would if the project were being implemented with a large, well-known and highly adopted CMS framework like WordPress or Drupal, or Sitepoint.

[52] http://en.wikipedia.org/wiki/Blue_Ocean_Strategy

Note: In this author's opinion, the learning curve necessary to acquire the skills required for the development of a modern website or web application tends to actually be the same, no matter whether you "start from scratch" or decide to use a large CMS framework. Even though learning a large CMS framework like Drupal, for example, may take several months in order to be productive, the same is equally true when it comes to getting up to speed with other frameworks. In the case of Drupal you may spend a lot of time learning "how things are done". But if you go the other route, you will be learning "how things are done" in a whole series of smaller frameworks; while needing to find out "how to do things" quite a bit more than you would have to in the former case. So, starting from scratch may take even longer. I would say it's six of one and half a dozen of the other. Furthermore, much of what "learning Drupal" is often said to entail actually includes things proper to the web application domain itself, such as server provisioning, e-commerce, email management, maintenance, testing frameworks, and methods of deployment, etc. So this isn't a pro or a con, again, in this author's opinion and experience.

"Frameworks don't scale!" Well, this is simply just not true. The Emmys, Weather.com, and many other examples abound. Of course, it's going to call for a lot of work to get a framework to scale. It's not the kind of work you expect to be doing when you signed up for a CMS framework either: Load balancers, Content Delivery Networks, HTTP and Database optimizations, accelerators, are just the start. Bud did you seriously think you were going to avoid all that just by using a lighter framework? Actually, it's all the same work. So this isn't a pro or con either.

In summary, reusability is an important principle and goal in software development, and for good reason. It is often summed up in the phrase "standing on the shoulders of giants". By observing this principle, one brings a necessary quota of zen modesty to the table and professionalizes ones activity. By admitting that existing CMS solutions have been perfected over time by their surrounding community of creators and users, and have honed issues such as security, functionality, ease of use, and many other fronts, developers gain power and productivity. There are CMS solutions for many different architectures and programming languages, and the best ones are fully customizable.

The reckless "hero" go-it-alone approach to software development, on the other hand, is often disguised improvisation, and a recipe for failure.

Proprietary or not? Open-source or not?

Once the choice of framework has been made for a given project or series of projects, it must then be decided to go either with a proprietary or an open source solution.

On this subject, let's first take a look at the title of a fairly recent article: ["Open Source vs. Proprietary Software: There is No Clear Winner"[(http://www.cmswire.com/cms/information-management/open-source-vs-proprietary-software-there-is-no-clear-winner-021752.php#null). This

might inflame the hearts and minds of many of us who have unquestionably supported open source over the years. But it's time to coldly ask ourselves: which has better code, and what's the low-down about true cost to market?

The article itself begins by asking why anyone would want to ever use anything but Open Source Software: "It's free (or almost free); it's built by passionate communities of developers; you can "look under the hood"; and there's no vendor lock-in. Add to that, that the rate of innovation is supposed to be faster." And the short answer is that after a certain point of complexity is reached, it ceases to be true: it's no longer free even if there is no actual price tag on it, since the cost of (professional) use is much higher; it's increasingly built by company employees rather than a broad based passionate community; the rate of innovation ceases to be fast; and a kind of vendor lock-in arises out of the sheer scale becoming necessary for development, deployment and maintenance.

- According to Software development testing leader Coverity[53], defect densities are practically identical. Where the number of lines of code[54] exceed 1 million, proprietary quality is better. The reason seems to be that not all open source projects have the budget to deal with the infrastructure required by the added complexity. Interestingly enough Drupal has 970,000[55], just under the limit.
- "The true value in open source is the open, collaborative community of like-minded professionals who come together to crowd-source ideas and build great things" (Kevin Cochrane, Alfresco). But proprietary software vendors have huge community support to, as is undeniably the case with Sharepoint Saturday's[56]. They have the "speakers", the "sponsors" and the T-shirts!
- In fact, there is a growing tendency within all open source frameworks for the number of very key core contributors to form part of an inner circle. In this way, the gap is closing between the relatively small number of "employee" committers on a proprietary project, on the one hand, and the ever-decreasing number of core committers on open source projects that grow in complexity.
- Cost to market actually converges in both cases also, even though of course results may vary: Proprietary solutions charge an up-front and/or licensing price, but Open Source frameworks require considerable development, training and administratation costs to accomplish what the former does out-of-the-box.

In this author's opinion technical support and infrastructre service levels may be obtained for both kinds of frameworks; and there are more and more consultants, agencies and companies with an enormously high level of domain knowledge for either kind of framework also. These areas cannot really be considered to be an advantage for only proprietary frameworks.

[53] http://www.coverity.com
[54] http://www.ohloh.net
[55] http://www.ohloh.net/p?ref=homepage&q=drupal
[56] http://www.spsevents.org

Rather than the choice being between Proprietary and Open Source, it is more about the project's real needs. "Paying developers to massively customise an open source CMS may end up costing you more than paying for a proprietary solution that more closely meets your requirements. On the flip side, paying license fees for an ill-suited proprietary solution doesn't make any sense either." (See Choosing Between an Open Source or Proprietary Website CMS[57]).

WordPress or not?

The stark fact here is that, from a CMS framework perspective, WordPress is less than Drupal, unless what you really want is an elegant industry-standard blog. While Drupal is a superset of WordPress, and one may migrate to Drupal comfortably without expecting to miss any features, the opposite is certainly not the case.

Unless your use cases coincide with the functionality offered by WordPress, in which case you still have a choice to make, the following points point consistently in Drupal's favor:

- It's when you contemplate porting a Drupal web application to a WordPress website that the latter's shortcomings really stand out.
- Despite Easy Post Types[58] Drupal wins hands down in terms of being able to create business objects with content types, having support for all kinds of field types, plus field collections and other features.
- Users are just another entity in Drupal and can use the same patterns that any old business object (taxonomy term, content type item, user, etc.) uses.
- Separation of concerns between the dynamic data model, the semantic HTML containing the dynamic content of the business object data model, and the styling to be applied to this basic unadorned un-styled HTML (that's called semantic HTML) is very well observed in Drupal, but badly supported in Wordpress, which means parrallel team development work is greatly impaired.
- WordPress has nothing like Drupal's Views: a complete and intuitive SQL Query Generator that's so intuitive most users don't realize it's an SQL Query Generator. It's really easy to list your stuff in all kinds of different ways.
- Rules module: built in support for UI based business rules, plus the ability to roll your own in code.
- Huge amount of very rich extensions (Drupal modules):
  - Complete layout and theming development system alternatives
    * Page Manager and Panels[59] (either with Panopoly[60] or do it yourself (as we do in our DurableDrupalDistro[61])
    * Display Suite[62]

[57] http://www.netxtra.net/insights/choosing-between-an-open-source-or-proprietary-website-cms/
[58] https://wordpress.org/plugins/easy-post-types/
[59] https://www.drupal.org/project/panels
[60] https://www.drupal.org/project/panopoly
[61] https://github.com/victorkane/drupal-lean
[62] https://www.drupal.org/project/ds

        * Context module[63]
    - Workflow modules
        * Workbench[64]
        * Workflow[65]
- Everything in code alternatives allow us to break the back of configuration in the database, and help our teamwork.

Drupal or not?

Drupal, together with its particularly rich eco-system, is a perfect fit for many content-centric projects because it offers such a complete set of solutions to the problems and challenges historically facing developers of dynamic websites and web application. Its features cover the whole cycle of design, development, deployment and maintenance:

- Dynamic website and web application framework with complete separation of concerns between business objects, data persistence, business logic and presentation.
- Social web platform
- Prototyping tool for Business objects and data modeling, data viewing, layout and styling
- Modern web design templating system allowing for further separation for concerns within the presentation layer, of semantic content, layout and styling.
- Built in and customizable Content Management and Editorial Workflow System
- Well-documented and established tools and procedures make for full compatibility with best practices on development process and devops levels.

Drupal sits under the 1 million lines of code limit, which would indicate that it is a project whose quality has greatly benefited over the years from having been developed by a diverse and large international community. Since the code is open source, it has been available to all to freely criticize, modify (or at least propose patches in issue queues), and discuss. It is in fact one of the most sophisticated CMS frameworks around, whether open source or proprietary, capable of supporting the development of large-scale content-centric web application projects.

Many of the benefits of Drupal stood out in the previous section when it was compared to WordPress. But there's more.

Community and Intelligence

Drupal has a truly international community, with a huge amount of communication, sharing of responsibility and participation of all kinds going on across borders at all times. Each year there are two international conferences (DrupalCon), one in Europe and one in the US. 2015 will see the first DrupalCon Latin America[66]. In addition, Drupaleros in many cities all over the world hold annual

[63] https://www.drupal.org/project/context
[64] https://www.drupal.org/project/workbench
[65] https://www.drupal.org/project/workflow
[66] https://latinamerica2015.drupal.org/

Drupal camps, some having a long and rich history such as the Bay Area's BADCamp[67]. There are many additional ways[68], online and off, that interaction takes place in the community. It can be said in many ways that technical expertise is part and parcel of the Drupal culture, as is the case in many open source communities.

And since the Drupal community is not based solely in a single country, but rather all over the world, the approach tends to be more cosmopolitan and "out of the box", with people looking for deep and permanent solutions and not simply quick fixes based on a "common sense" empirical ways of thinking. That's why even though Drupal could have been written off quickly with the advent of Server side java script and a bevy of front-end frameworks (as of course it could be one day), or perhaps by new offerings from other programming environments and architectures, a huge critical mass of intelligence has forged Drupal into something truly modern. This promises years of competitive mojo: separation of concerns with adaptive back and front-ends, straightforward integration with multi-tiered and distributed components, REST interfaces both for stand-alone servers and clients.

Easily extensible

Modules

Since highly talented people have always been attracted to the community, solutions have constantly emerged keeping those leveraging Drupal for their content-centric projects on the technological vanguard with minimal effort. In a rich, mature eco-system, there is, literally, "a module for everything", and not just those with sexy features, but also those promising integration with all major players in the modern web app universe. Today, for example, with no coding you can whip up a fully functional social network based platform with a back-end API, with node.js based chat and activity stream, and with a totally mobile first responsive theme to top it off in literally hours.

Distributions

In addition many first-class Drupal installation profiles ("distros") are available. These off-the-shelf domain-specific customizable solutions tailor Drupal to specific project needs. Some of the best examples:

- CiviCRM Starter Kit[69]: a web-based, open source, CRM geared toward meeting the needs of non-profit and civic-sector organizations. The starter kit integrates CiviCRM with Drupal.
- [DKAN[(https://github.com/NuCivic/dkan): a Drupal-based open data platform with a full suite of cataloging, publishing and visualization features that allows governments, nonprofits and universities to easily publish data to the public.

[67] https://badcamp.net/

[68] https://www.drupal.org/community

[69] https://www.drupal.org/project/civicrm_starterkit

- Commerce Kickstart[70]: a software distribution for sites of any size that provides all the power and flexibility of Drupal Commerce, combined with components that accelerate creation and launch of an online store.
- Open Academy[71]: a web publishing solution for higher education.
- OpenAid[72]: a turnkey website platform designed to help cause-driven organizations create cost-effective program-focused websites quickly.
- Open Atrium 2[73]: open source collaboration software that enables organizations to securely connect their teams, projects, and knowledge.
- OpenIdeaL[74]: an idea management system, for both public and commercial sectors. It is a powerful tool to analyze public opinion regarding products/services and identify trends.
- Open Outreach[75]: a Drupal distribution written to provide grassroots, activist, and nonprofit groups with the web tools they need for effective public engagement.
- OpenPublic[76]: provides a content management system specially designed for open government goals, without compromising accessibility, security or usability.
- Panopoly[77]: an off-the-shelf responsively themed, Panels based distribution designed to be a base framework upon which to build other Drupal distributions (among others, OpenAcademy and Open Atrium), but will also work for general site building.
- Plato Típico[78]: This is a a preconfigured distro with multi-language support for English & Spanish. Already comes with most of the necessary modules which will save you lots of time and research.
- Pushtape[79]: Build better music websites with Pushtape, a Drupal distribution for musicians. Manage your music discography, create news updates, upload photos, post upcoming shows and more.

Many may be installed with a single click on the Pantheon[80] platform, for example.

Themes

The Drupal theme system is everything that the superficial and outmoded industry standard "template" concept is not. Themes allow for the meticulous separation of semantic data and business logic, on the one hand, from the layout, styling and formatting of the presentation layer, on the other. Characteristic of Drupal, they provide compelling alternatives for design-level customization.

[70] https://www.drupal.org/project/commerce_kickstart
[71] https://www.drupal.org/project/openacademy
[72] https://www.drupal.org/project/openaid
[73] https://www.drupal.org/project/openatrium
[74] https://www.drupal.org/project/idea
[75] https://www.drupal.org/project/openoutreach
[76] https://www.drupal.org/project/openpublic
[77] https://www.drupal.org/project/panopoly
[78] https://www.drupal.org/project/plato_tipico
[79] https://www.drupal.org/project/pushtape
[80] https://www.getpantheon.com/

Recognizable, standard architecture

If you build it with Drupal, automatically an increasing number of readily available Drupal development talent is available all over the world who will recognize how you web app is built, and be able to provide maintenance or further development as you wish. Compared to the effort required for a developer to comprehend the anatomy of a one-off custom application, that is a decided advantage and greatly enhances the lifetime value of your project.

An abundance of Drupal-friendly deployment options

Depending on the scale of what you are deploying, there are a host (pardon) of infrastructure alternatives: From shared hosting (yes, it's gotten better), to VPS and cloud PaaS, including the Pantheon platform mentioned above. It's getting easier and easier for Drupal based web applications to enjoy smooth delivery and maintenance.

So much internationally active talent, so much engineering, so much quality! Drupal excels as a CMS and Web Application Framework.

Drupal 8 or not?

Drupal 8 cannot be used now

> "Drupal 8 is currently in development and is not yet ready for production use." > *drupal.org*[81]

> "Drupal 8 will set a new standard for ease of use, while offering countless new ways to tailor and deploy your content to the Web. Easily customize data structures, listings, and pages, and take advantage of new capabilities for displaying data on mobile devices, building APIs, and adapting to multilingual needs.
>
> "With a leaner and meaner core, easier migration process from earlier versions, in-place content editing tools and loads more power for modules and themes thanks to a modern Object Oriented Programming (OOP) approach on the backend, there is something for everyone to love in Drupal 8. When will it be released? See the FAQ[82]." > drupal.org on Drupal 8[83]

So however interesting Drupal 8 and Symfony may appear cannot be used now for production website or web applications.

[81] https://www.drupal.org/documentation/version-info
[82] https://drupal.org/drupal-8.0/faq
[83] https://www.drupal.org/drupal-8.0

Drupal 8 will be ready for non-trivial site building sometime in 2016

Drupal Planet[84] consensus seems to indicate that Drupal 8 will probably not have a production ready release until around mid-2015, and that another 6-12 months will have to go by before:

- The offering becomes stable.
- Most third party modules and distributions get ported to Drupal 8, if at all, or replaced.
- A critical mass of Drupal developers become profficient in Drupal 8 development and it becomes cost-effective to equip a team with available developers.

This is somewhat mitigated by the adoption in Drupal 8 core itself of key functionality that has historically been relegated to third-party modules:

- Configuration management
  - Based on versionable YAML text files, no longer mixed with content in the database
- i18n and l10n multilingual platform
- Content listings
  - Views api and UI
- More field types
- The adoption of OOP and a rewritten core bring a number of advantages on the basis of its refactored routing component
  - The Drupal 7 context module is now in core
  - Web services *Think REST API or services back-end option
  - Block is just another entity
- HTML 5 markup
- Twig template based theming
- Editorial benefits in core
  - In place editing
  - WYWIWYG editing in core
  - (Some feel these kinds of features undermine structure content)

But you will have to calculate true development costs since third-party based functionality you take for granted today may not be available.

[84] https://www.drupal.org/planet

Straight up Symfony or not?

Why not just go with Symfony then and just skip the Drupal Drama? We owe it to ourselves to consider this alternative coldly and professionally. Our question has to be, just what are the merits of website and web application development just going with Symfony?

But using Symfony "instead" of Drupal is roughly equivalent to Starting from Scratch: Either you replace Drupal yourself individually with your own development resources (pretty difficult) or you replace it with another framework.

Symfony being a PHP based framework is going to be an advantage for some and a disadvantage for others. On the one hand it leverages the skills you have gained with Drupal or other frameworks without having to learn a new language or actually set of front-end and back-end languages (javascript, go, java/scala, ruby, etc.). On the other, some developers who are looking to grow beyond PHP, or feel it is limited in some way[85], if they are going to go with a lighter framework would prefer to go with one of the non-PHP alternatives.

Backdrop or not?

Backdrop was forked from Drupal as a reaction against Drupal 8 for the following reasons:

The Roadmap for Backdrop 1.0[86] lists the following:

- Configuration Management System
- Built-in Content Listing Generator
- Revamped Layout System
- Improved Performance
- Improved Mobile Support
- Reduced Theme System Complexity

How well will these be implemented?

Will it attain community critical mass?

Will it think out a proper workflow (i.e. be a solution for devs as well as end users)? The most important Backdrop feature is workflow with everything in code. D8 already has their workflow. What's Backdrop's? Difficulties in getting onto Pantheon for example, show workflow not thought out. For example, export and import of config from one system to another is easy in the GUI. But what if I want to automate that with ansible as part of a deployment scheme? In my opinion, we would want code+db+files+config, we don't want config to be mixed either with code for all sites (since it is specific to one) … we don't want you to have to push files or push database to live!!!!!!!

[85] http://eev.ee/blog/2012/04/09/php-a-fractal-of-bad-design/

[86] https://backdropcms.org/roadmap

We'll come back to this (see how it matures) in the Migrations volume (next) in the DurableDrupal series. (we'll fork backdrop, practice using it, then make a push request or else at the very least raise an issue concerning workflow blockers when we write Vol. 2 (Migrations) for the DurableDrupal series.

Drupal 7 or not?

State of Drupal 7

Drupal 7 is mature, full of add-ons, modules, distros, and quality base themes many of which offer an out-of-the-box responsive front-end. There is a huge community of skilled designers and developers thoroughly experienced in the release, so building a team is much more accessible than most other options. And Drupal 7 will still be officially supported for another three years or so, and then probably for some time after that, at least informally (as has been the case with other releases).

It's available right now as an option for including much of the hotness of Drupal 8 without the learning curve. Given the complete panorama we have been examining, Drupal 7 is a high-quality, low-cost alternative that should definitely be considered first, and only discarded if there exist unusual requirements on a given project.

Now, this very maturity and rich flexibility, plus the breadth and experience gained in the community with Drupal 7 gives us an exciting perspective: the makings of a Drupal LTS distro we can build on now and maintain for years to come.

Why a distribution?

To understand the importance of basing ourselves on a Drupal distribution, or installation profile, let's review the procedures that would normally be followed in the context of firing up a new project:

1. Prepare a development instance on a LAMP stack somewhere with a database and a runnable document root
2. Run the standard Drupal 7 install profile on that development instance
3. Get rid of the irritating cruft
4. Decide which responsive layout systems to include (Drupal core blocks and Context[87], Panels[88], Display Suite[89])
5. Create a sub-theme from the base theme of your choice (i.e. AdaptiveTheme[90], Zen[91], Omega[92], just to name three) compatible with your layout systems

[87] https://www.drupal.org/project/context
[88] https://www.drupal.org/project/panels
[89] https://www.drupal.org/project/displaysuite
[90] https://www.drupal.org/project/adaptivetheme
[91] https://www.drupal.org/project/zen
[92] https://www.drupal.org/project/omega

6. Install the third-party contrib modules without which you cannot build a site and which our experience has told us are good solid building blocks (80/20 rule[93])

Then, when you want to fire up a separate staging instance for acceptance testing with the client, you:

1. Prepare a development instance on a LAMP stack somewhere with a database and a runnable document root
2. Copy over the code (Drupal) and the files (assets)
3. Dump the dev database and populate the staging database
4. Edit settings.php
5. Clear cache :)

A lot of the bother can be solved, it's true, using the famous "everything in code" paradigm, especially for continuous delivery to staging from dev, and we'll see how that's done later:

1. Create a repo on GitHub/Bitbucket or similar
2. Export the current state into versionable text files (we'll see how a bit later on) and push to the repo
3. Clone this master dev site from the repo to staging
4. Clear cache :)

What if we could automate this process, eliminating the need to execute these steps manually, over and over again, every time we fire up a new project?

What would we need to be able to do that?

Answer: An on-going in-house Drupal distribution. Together with a lean agile process (skip ahead), a repo-based everything-in-code development discipline (skip ahead), and a DevOps[94] automation platform for provisioning, deployment, and orchestration (like Puppet[95], Chef[96], Ansible[97]: skip ahead).

That's why a distribution must form the heart of our process.

[93] http://en.wikipedia.org/wiki/Pareto_principle#In_software
[94] http://en.wikipedia.org/wiki/DevOps
[95] http://puppetlabs.com/
[96] https://www.chef.io/chef/
[97] http://www.ansible.com/home

Presenting DurableDrupalDistro with Lean process

This book will allow us to tool up a website and web app factory process on the solid basis of forking, using, and repeating our best practices as we re-use the DurableDrupalDistro.

Whether we are talking about working on our local workstation or laptop, on a PaaS (Platform as a Service) like Pantheon or Platform.sh, or whether we are going to be using a VPS or cloud instance of our own, the secret is to automate DevOps on the basis of our very own Drupal distribution. This all-important in-house Drupal distribution, will evolve with us from project to project and become part of an organization's library of distributions as teams get formed around new projects. It will be branched and tailored (with the diff merged back in, wholly or partially, in order to benefit future projects). And it will be the stepping stone to migrations to other Drupal releases, to Symfony, or any other future framework the future may hold for us. We will be able to build a development process of best practices around it, and set up our own veritable web factory.

Fork it, tailor it, maintain it, branch and release it over all your projects!

Evolve it, build a team process around it and grow it into the future!

In this chapter, given a project we need to get started on, a dynamic website for a neighborhood non-profit that runs programs for youth in the community, we've laid all our options face-up on the table, we've taken a decision on what's best to use right now. We reviewed starting from scratch, proprietary and open-source frameworks, WordPress and Drupal, Drupal 8, Backdrop CMS, Drupal 7, and, given Drupal 7, the use of a distribution, the DurableDrupalDistro, as a starting point upon which to build.

In the next chapter we'll take the DurableDrupalDistro and run with it and build our project.

Appendix 3. Team Workflow and Provisioning

Team roles

Basing ourselves on Web Style Guide by Patric J. Lynch and Sarah Horton[98], published a few years ago in 2009 by Yale University Press. At that point, the traditional Website development team core skillset were presented as being:

- Strategy and planning
- Project management
- Information architecture and user interface design
- Graphic design for the web
- Web technology
- Site production

with Web Team roles and responsibilities being:

- Project stakeholder or sponsor
- Web project manager
  - Account executive
  - Quality assurance tester
- Usability lead
- Information architect
- Art director
  - Web graphic designer
  - Interactive designer (Flash, JavaScript, Ajax)
  - Media specialist (photography, illustration, audiovisual, Adobe Flash)
- Web technology lead
  - Web application programmer (.Net, Java, php/Perl, Ruby)
  - Web page engineer (xhtml, css, JavaScript, Ajax)
  - Database administrator
  - Web systems expert or webmaster
- Site production lead

[98] http://webstyleguide.com/wsg3/1-process/2-development-team.html

    - html page coder
- Site editor
- Site copywriter
- Content domain expert (content coordination, research)

TODO Translate this in steps to our current team role set introduced in Chapter 1.

Basic team workflow

clone pull, branch, work, pull request rinse and repeat

"We'll first show you how to set up a workspace[99], getting a folder somewhere you can configure as a Drupal doc root, that is, the main Drupal directory that can be picked up by a LAMP stack and made available to a browser as a URL.

This can be done on a dedicated server, on a VPS, on a virtual machine or directly on your laptop, even on a good shared hosting, but you've got to have ssh access to the command line.

We all need to be able to do this no matter what our role on the team, so we'll show you a couple of examples.

Then we'll see how to grab our very own DurableDrupalDistro and fire it up[100].

Finally, we'll check out what we have going for us[101] under the hood right off-the-shelf, we'll take our first steps[102], and then I'll give an example of updating and maintaining the distro itself[103]".

Getting a workspace in a LAMP stack

El Viejo shows here how to set up a workspace from scratch, in his usual opinionated fashion.

- on a VPS like Linode or Digital Ocean,
- on personal laptop/workstation alternatives
    - Water Works stack
    - Bitnami
        * Drupal App
        * Drupal Server
    - Kalabox
- Roll your own
- on a shared hosting account with command line access and drush

[99] a_workspace_in_a_lamp_stack.html
[100] obtaining_and_installing_the_durabledrupaldistro.html
[101] whats_included_in_durabledrupaldistro.html
[102] durabledrupaldistro_first_steps.html
[103] updating_durabledrupaldistro_itself.html

VPS

TODO linode

TODO digital ocean

Water Works stack

Well, Water Works never got around to offering one, or else they did and then started charging for it, so nobody used it and it died. But Acquia has one! Very convenient for running Drupal apps on your laptop, but it isn't a virtual machine, it's an app only, so that means that while you can run it and everything, it doesn't emulate the target live environment (a GOOD idea) the app will be running on when it is launched. Get it here[104].

Bitnami

Bitnami[105] swings both ways, with the app approach and the VirtualBox VM alternative (recommended but a lot of work[106] and then more work[107]).

Pantheon

Single click .....

And you can have the best of both worlds, with pantheon in the cloud and on your desk... Kalabox!

Kalabox

Ah, the amigos at Kalamuna[108] are a breath of fresh air. They've not only come up with Kalabox[109] but are now developing Kalabox 2[110], the NodeWebkit[111] and Docker[112] wonder, as a completely open source project after a successful Kickstarter campaign[113].

I've tried it out quite a bit. See how I installed Kalabox 1[114] and used it as a Pantheon workflow solution. Integration with Pantheon hosting[115] and their one-click Drupal Distro site builder is a

[104] http://www.acquia.com/downloads
[105] https://bitnami.com/stack/drupal
[106] http://awebfactory.com/node/524
[107] http://awebfactory.com/node/525
[108] http://www.kalamuna.com/
[109] http://www.kalamuna.com/products/kalabox
[110] https://github.com/kalabox
[111] https://github.com/rogerwang/node-webkit
[112] https://www.docker.com/
[113] https://www.kickstarter.com/projects/kalabox/kalabox-advanced-web-tools-for-the-people
[114] http://awebfactory.com/node/522
[115] https://www.getpantheon.com/

hallmark of Kalabox 1. Very useful considering its revolutionary architecture[116] and standardizing workflow[117] (at least as far as Drupal hosting is concerned). Alternative hosting support plug-ins are expected with Kalabox 2.

Ansible

(study up on Ansible for Dev-Ops book first! buy the bundle!) * Ansible on a Mac to run a server * Ansible on a Mac to run a local vagrant/vbox instance

Roll your own

sthg about virtual box & vagrant, with puppet, chef, ansible…

Shared hosting account

We shouldn't even be talking about this, but recent improvements in shared hosting accounts have made something like a reseller account somewhat (only just) viable, at least for hosting small to middling projects.

First thing to do is get Drush on there, once you've gotten ssh command-line access, if you can't get that switch hosting now (and if you're even reading this section, second thing is to create a subdomain for each site: don't go running Drupal from a sub-directory).

Bringing the whole team up with the DurableDrupalDistro

What's included in DurableDrupalDistro

DurableDrupalDistro for our first project

So we're in oue workspace, let's say on a VPS.

What we want to do

- Create virtual host
- Create database
- Clone and set up distro
- Install with drush on the command line
- Check it out

[116] https://www.getpantheon.com/how-it-works
[117] https://www.getpantheon.com/how-it-works

Create virtual host

Create database

Clone and set up distro #### Install with drush on the command line #### Check it out

Updating DurableDrupalDistro itself

```
$ drush pm-refresh
Refreshing update status information ...
Done.
$ drush pm-update
 Name                 Installed  Proposed  Message
                      Version    version
 Drupal               7.31       7.32      SECURITY UPDATE available
 Calendar (calendar)  7.x-3.4    7.x-3.5   Update available
 CKEditor (ckeditor)  7.x-1.15   7.x-1.16  SECURITY UPDATE available
 Profiler Builder     7.x-1.1    7.x-1.2   Update available
 (profiler_builder)


Update information last refreshed: Tue, 10/21/2014 - 13:20
NOTE: A security update for the Drupal core is available.
Drupal core will be updated after all of the non-core projects are updated.

Security and code updates will be made to the following projects: Calendar [cale\
ndar-7.x-3.5], CKEditor - WYSIWYG HTML editor [ckeditor-7.x-1.16], Profiler Buil\
der [profiler_builder-7.x-1.2]

Note: A backup of your project will be stored to backups directory if it is not \
managed by a supported version control system.
Note: If you have made any modifications to any file that belongs to one of thes\
e projects, you will have to migrate those modifications after updating.
Do you really want to continue with the update process? (y/n):
Project calendar was updated successfully. Installed version is now 7.x-3.5.
Backups were saved into the directory                                 [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/modules/calendar.
Project ckeditor was updated successfully. Installed version is now 7.x-1.16.
Backups were saved into the directory                                 [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/modules/ckeditor.
Project profiler_builder was updated successfully. Installed version is now 7.x-\
1.2.
```

```
Backups were saved into the directory                              [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/modules/profiler_build\
er.

Code updates will be made to drupal core.
WARNING:  Updating core will discard any modifications made to Drupal core files\
, most noteworthy among these are .htaccess and robots.txt.  If you have made an\
y modifications to these files, please back them up before updating so that you \
can re-create your modifications in the updated version of the file.
Note: Updating core can potentially break your site. It is NOT recommended to up\
date production sites without prior testing.

Do you really want to continue? (y/n):
Project drupal was updated successfully. Installed version is now 7.32.
Backups were saved into the directory                              [ok]
/home/drupallean/drush-backups/drupal_lean/20141021132817/drupal.
No database updates required                                       [success]
'all' cache was cleared.                                           [success]
Finished performing updates.                                       [ok]
$ git status
$ git add .
$ git commit -am "Maintenance update to drupal core 7.34 plus updated contrib mo\
dules"
$ git push
$ git tag
$ git tag -a "December2014" -m "Maintenance core and contrib update"
$ git push --tags
```

Now we need to prepare the site (see Quick install for developers (command line)[118]

```
cp sites/default/default.settings.php sites/default/settings.php
```

Give the web server write privileges (666 or u=rw,g=rw,o=rw) to the configuration file.

```
chmod a+w sites/default/settings.php
```

Give the web server write privileges to the sites/default directory.

[118] https://www.drupal.org/documentation/install/developers

```
chmod a+w sites/default
```

so we can re-install from scratch to test. Check out incredible options available with drush help:

```
$ drush help site-install
Install Drupal along with modules/themes/configuration using the specified
install profile.

Examples:
 drush site-install expert --locale=uk     (Re)install using the expert install
                                           profile. Set default language to
                                           Ukranian.
 drush site-install                        Install using the specified DB
 --db-url=mysql://root:pass@localhost:por  params.
 t/dbname
 drush site-install                        Install using SQLite (D7+ only).
 --db-url=sqlite://sites/example.com/file
 s/.ht.sqlite
 drush site-install --account-name=joe     Re-install with specified uid1
 --account-pass=mom                        credentials.
 drush site-install standard               Pass additional arguments to the
 install_configure_form.site_default_coun  profile (D7 example shown here - for
 try=FR                                    D6, omit the form id).
 my_profile_form.my_settings.key=value
 drush site-install                        Disable email notification during
 install_configure_form.update_status_mod  install and later. If your server
 ule='array(FALSE,FALSE)'                  has no smtp, this gets rid of an
                                           error during install.

Arguments:
 profile                                   the install profile you wish to run.
                                           defaults to 'default' in D6,
                                           'standard' in D7+
 key=value...                              any additional settings you wish to
                                           pass to the profile. Fully supported
                                           on D7+, partially supported on D6
                                           (single step configure forms only).
                                           The key is in the form [form
                                           name].[parameter name] on D7 or just
                                           [parameter name] on D6.

Options:
```

```
 --account-mail                      uid1 email. Defaults to
                                     admin@example.com
 --account-name                      uid1 name. Defaults to admin
 --account-pass                      uid1 pass. Defaults to a randomly
                                     generated password. If desired, set
                                     a fixed password in drushrc.php.
 --clean-url                         Defaults to 1
 --db-prefix                         An optional table prefix to use for
                                     initial install.  Can be a key-value
                                     array of tables/prefixes in a
                                     drushrc file (not the command line).
 --db-su=<root>                      Account to use when creating a new
                                     database. Must have Grant permission
                                     (mysql only). Optional.
 --db-su-pw=<pass>                   Password for the "db-su" account.
                                     Optional.
 --db-url=<mysql://root:pass@host/db> A Drupal 6 style database URL. Only
                                     required for initial install - not
                                     re-install.
 --locale=<en-GB>                    A short language code. Sets the
                                     default site language. Language
                                     files must already be present. You
                                     may use download command to get
                                     them.
 --site-mail                         From: for system mailings. Defaults
                                     to admin@example.com
 --site-name                         Defaults to Site-Install
 --sites-subdir=<directory_name>     Name of directory under 'sites'
                                     which should be created. Only needed
                                     when the subdirectory does not
                                     already exist. Defaults to 'default'

Aliases: si
```

So let's install in Spanish, specifying db credentials, uid1 user name and password, uid1 user email, site email and site name

```
site-install drupallean --locale=es --db-url=mysql://drupal_lean:lean@localhost/\
drupal_lean --account-name=admin --account-pass=omg --account-mail=victorkane@gm\
ail.com --site-mail=drupallean@awebfactory.com.ar --site-name=DrupalLean

You are about to DROP all tables in your 'drupal_lean' database. Do you want to \
continue? (y/n): y
No tables to drop.                                                 [ok]
Starting Drupal installation. This takes a few seconds ...         [ok]
WD php: Warning: Invalid argument supplied for foreach() in        [warning]
_features_restore() (line 966 of
/home/drupallean/drupal-lean/sites/all/modules/contrib/features/features.module).
Installation complete.  User name: admin  User password: omg     [ok]
```