

Nearly Headless Drupal

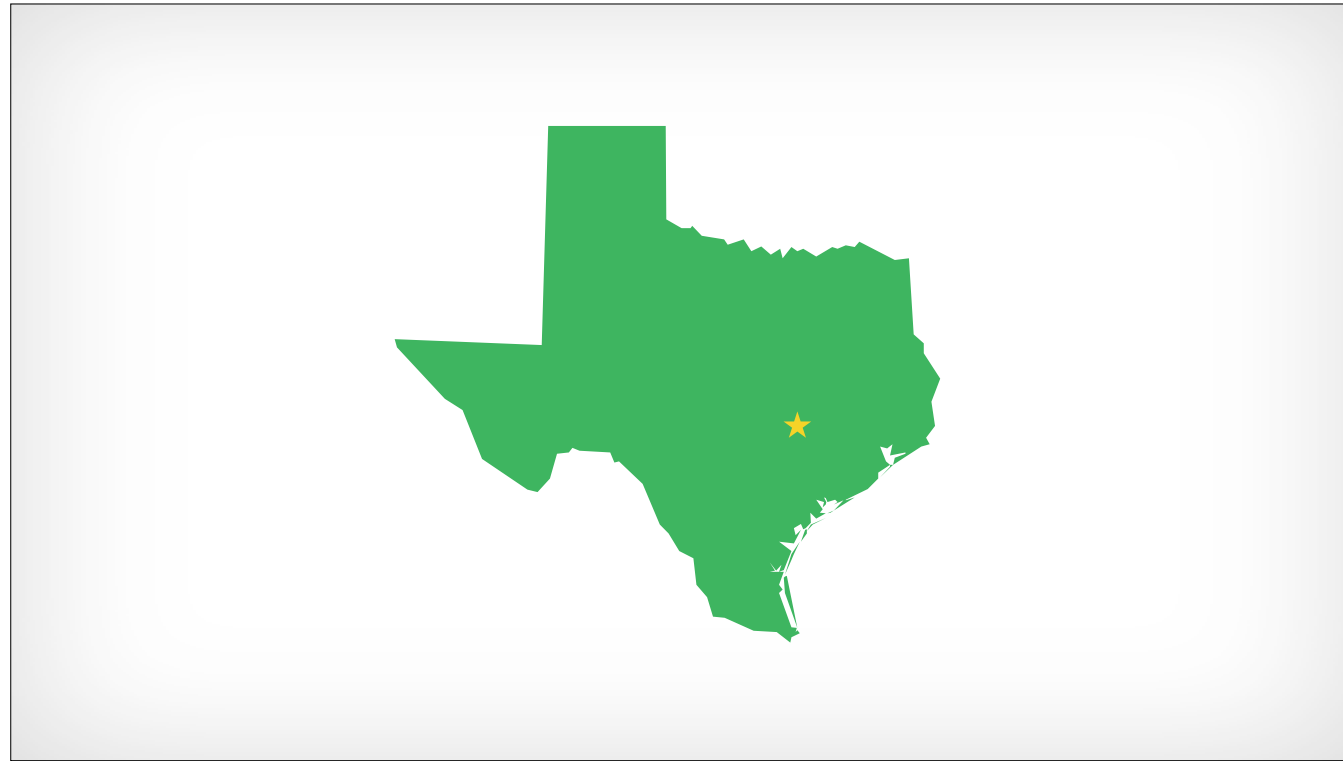
SANDCamp 2/26/2016



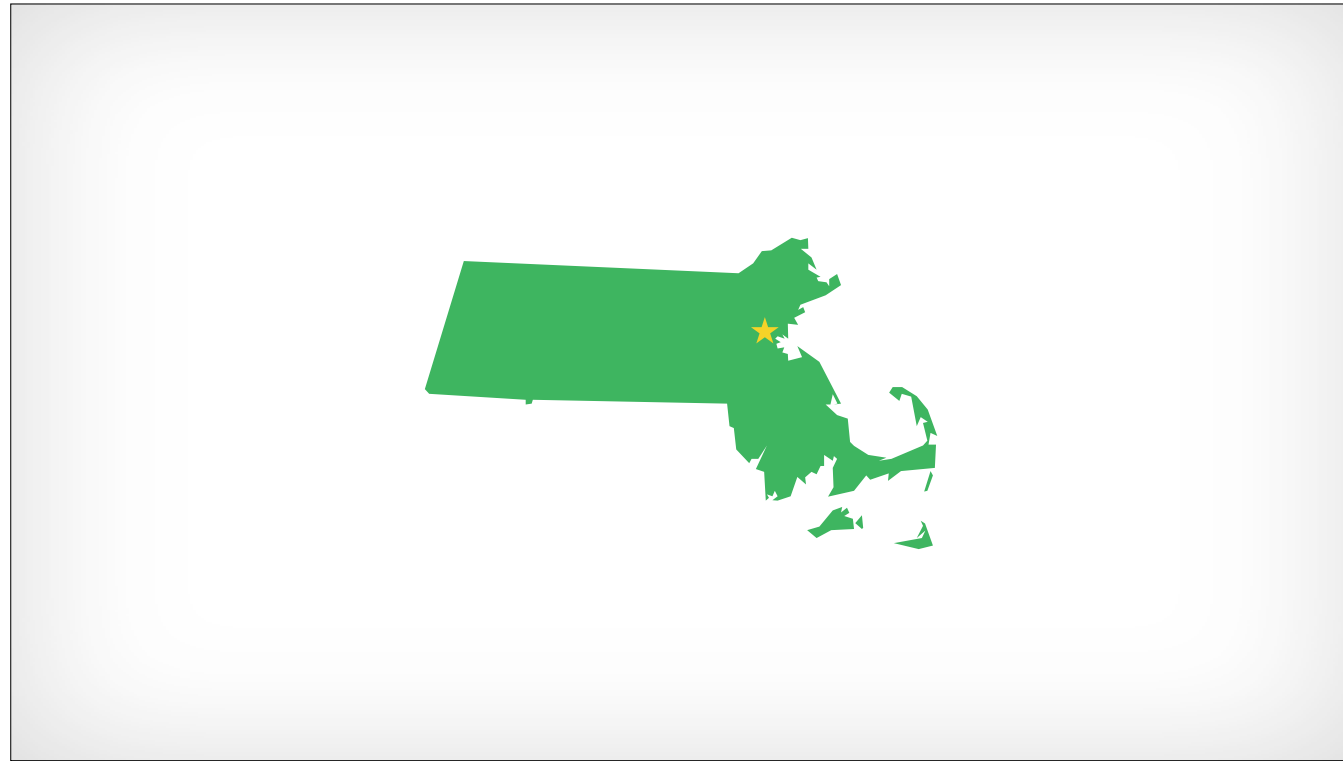
Howdy!
I'm an engineer at
Four Kitchens

Dustin Younse
@milsyobtaf

We're a Drupal shop based out of Austin, TX, where we have had the opportunity to work on some pretty interesting headless projects.



I'm originally from Austin



But now I hang my Stetson in Somerville, MA

What Is Headless Drupal?



what is headless drupal? well, first and foremost:

🔥 HOT DRAMA 🔥

🔥 HOT DRAMA 🔥

- <http://dgo.to/2645250>
- <http://buytaert.net/the-future-of-decoupled-drupal>
- <https://www.chapterthree.com/blog/why-progressive-decoupling-drupals-front-end-bad-idea>
- <http://www.marcdrummond.com/posts/2016/02/23/js-future-drupal-front-end-chat-with-socrates>

When I last gave this talk, at NERDSummit last September, headless / progressive was still a fairly quiet kind of thing. And then.

But this talk isn't about any of that. Seriously.

Why Headless?

- Content Portability
- Front End Flexibility
- Ease of upgrade / redesign
- Security (Kinda)

Content Portability – Site to site, site to app, TWiT example

Front end flexibility – you can hire domain specific people, rather than Drupal generalists. a lot of these issues are cleared up in d8, but not all

Ease of upgrade / redesign – you only need to upgrade one part of the system rather than both, and you can do this in a somewhat invisible manner, since you don't have to take down the existing infrastructure

Security –

Security, Kinda

www.sxbx.com/CHANGELOG.txt

```
Drupal 7.39, 2015-08-19
-----
- Fixed security issues (multiple vulnerabilities). See SA-CORE-2015-003.

Drupal 7.38, 2015-06-17
-----
- Fixed security issues (multiple vulnerabilities). See SA-CORE-2015-002.

Drupal 7.37, 2015-05-07
-----
- Fixed a regression in Drupal 7.36 which caused certain kinds of content types
  to become disabled if they were defined by a no-longer-enabled module.
- Removed a confusing description regarding automatic time zone detection from
  the user account form (minor UI and data structure change).
- Allowed custom HTML tags with a dash in the name to pass through filter_xss()
  when specified in the list of allowed tags.
- Allowed hook_field_schema() implementations to specify indexes for fields
  based on a fixed-length column prefix (rather than the entire column), as was
  already allowed in hook_schema() implementations.
- Fixed PDO exceptions on PostgreSQL when accessing invalid entity URIs.
- Added a sites/all/libraries folder to the codebase, with instructions for
  using it.
- Added a description to the "Administer text formats and filters" permission
  on the Permissions page (string change).
- Numerous small bug fixes.
- Numerous API documentation improvements.
- Additional automated test coverage.
```

One of the keys to security is reducing your attack footprint, hiding or reducing the number of places an attacker can poke and prod.

With a non-Drupal front end, you obfuscate the fact that your site is running Drupal. It's hardly 100% effective, but it makes you that much harder of a target to attack.

Security, Kinda

- /user/login
- /user/reset
- /node/add

This also removes an attacker's ability to go after

Headless Successes

- The Tonight Show with Jimmy Fallon
 - <http://www.nbc.com/the-tonight-show>
- This Week In Tech
 - <http://twit.tv>
- The Weather Channel
 - <http://www.weather.com>

Tonight Show – done by us in conjunction with Lullabot, the design firm CP+B, and the team at NBC Digital
twit.tv – done by us, top to bottom
[weather.com](http://www.weather.com) – done by our friends at Mediacurrent

Why Not Headless?



It's important to make sure you are building a website for the right reasons, using the right tools for the job. It's easy to get carried away with the latest new tech and lose sight of the product users will want to use. This isn't really a knock against headless, but a knock against doing headless wrong. (USA Today)

Why Not Headless?

- Complexity of Code
- Complexity of Hosting
- Reliance on non-Drupal code and infrastructure

You no longer have one repo – you now have a minimum of 2, and its easy to get more complex than that.
You no longer have one host – you now have a minimum of 2, and its easy to get more complex than that. (TWiT has three separate hosting bills now)
Reliance on third party js libraries or other non Drupal chunks of code or infrastructure



Nearly Headless?

Maybe we can try for a middle ground approach?

Use Cases

- Limited hosting resources

There are still plenty of reasons you would want to use a not-quite headless approach. Dipping your toe into a potential headless switch while using your currently limited hosting resources (Universities that must host in-house)

The URL Came From Inside The Site

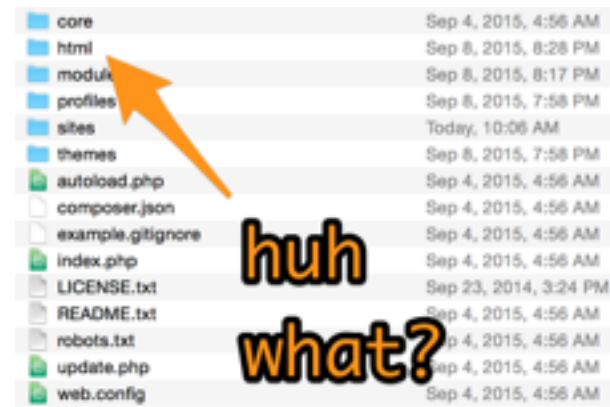
When you're dealing with Drupal, or any other CMS, you can sometimes lose sight of the forest for the trees. You have to remember that at the end of the day, you're just dealing with a web server, and web servers have some nice features

The URL Came From Inside The Site

core	Sep 4, 2015, 4:56 AM
html	Sep 8, 2015, 8:28 PM
modules	Sep 8, 2015, 8:17 PM
profiles	Sep 8, 2015, 7:58 PM
sites	Today, 10:06 AM
themes	Sep 8, 2015, 7:58 PM
autoload.php	Sep 4, 2015, 4:56 AM
composer.json	Sep 4, 2015, 4:56 AM
example.gitignore	Sep 4, 2015, 4:56 AM
index.php	Sep 4, 2015, 4:56 AM
LICENSE.txt	Sep 23, 2014, 3:24 PM
README.txt	Sep 4, 2015, 4:56 AM
robots.txt	Sep 4, 2015, 4:56 AM
update.php	Sep 4, 2015, 4:56 AM
web.config	Sep 4, 2015, 4:56 AM

Here is the base Drupal 8 folder structure

The URL Came From Inside The Site

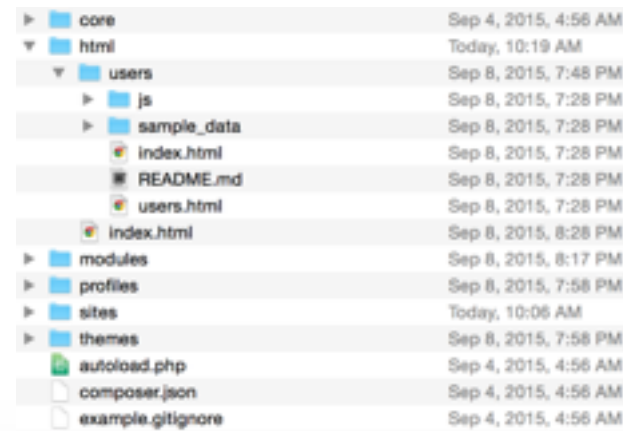


core	Sep 4, 2015, 4:56 AM
html	Sep 8, 2015, 8:28 PM
modules	Sep 8, 2015, 8:17 PM
profiles	Sep 8, 2015, 7:58 PM
sites	Today, 10:06 AM
themes	Sep 8, 2015, 7:58 PM
autoload.php	Sep 4, 2015, 4:56 AM
composer.json	Sep 4, 2015, 4:56 AM
example.gitignore	Sep 4, 2015, 4:56 AM
index.php	Sep 4, 2015, 4:56 AM
LICENSE.txt	Sep 23, 2014, 3:24 PM
README.txt	Sep 4, 2015, 4:56 AM
robots.txt	Sep 4, 2015, 4:56 AM
update.php	Sep 4, 2015, 4:56 AM
web.config	Sep 4, 2015, 4:56 AM

huh
what?

Wait, that isn't a default folder

The URL Came From Inside The Site



core	Sep 4, 2015, 4:56 AM
html	Today, 10:19 AM
users	Sep 8, 2015, 7:48 PM
js	Sep 8, 2015, 7:28 PM
sample_data	Sep 8, 2015, 7:28 PM
index.html	Sep 8, 2015, 7:28 PM
README.md	Sep 8, 2015, 7:28 PM
users.html	Sep 8, 2015, 7:28 PM
index.html	Sep 8, 2015, 8:28 PM
modules	Sep 8, 2015, 8:17 PM
profiles	Sep 8, 2015, 7:58 PM
sites	Today, 10:06 AM
themes	Sep 8, 2015, 7:58 PM
autoload.php	Sep 4, 2015, 4:56 AM
composer.json	Sep 4, 2015, 4:56 AM
example.gitignore	Sep 4, 2015, 4:56 AM

But it is a totally valid web folder, full of regular web content. Anything that shows up here will be accessible through your website. It's not pretty, and it will be overridden by any aliases you have in Drupal, but it works.

Use Cases

- Limited hosting resources
- Prototyping
- Incremental upgrade plan
- Adding external features

There are still plenty of reasons you would want to use a not-quite headless approach.

Prototyping a new front end using your existing data store.

Dipping your toe into a potential headless switch while using your currently limited hosting resources
(Universities that must host in-house)

Preparing for an incremental upgrade.

Adding external features – mobile apps, information kiosks (NYCCamp UN example), internal dashboards



Something like this (<http://www.panic.com/statusboard/>) can be powered entirely by the JSON your Drupal site spits out

Drupal 8 Makes It Easy

Drupal 8 Makes It Easy(ish)

- Views now in core
- REST Exports now in core
- New Javascript now in core
- Adding 3rd Party Javascript easier than ever

API Basics

- APIs as a contract
 - Versioning
- Create an interface, not an implementation
- API Design, The Musical, David Diers

Unless you are creating something that will only ever be consumed by your own personal applications (and you are willing to accept your teeth gnashing in the future as you curse the house of your former self) treat your API as a contract. Once it's in the wild, it should not change.

Of course, you will obviously have to change some things, and that's where versioning comes in. Views doesn't make this super easy, but it's manageable.

When you are creating your API, you want to expose “interface” type names (name, project_name, user_image) not “implementation” names (field_user_name, field_project_name_2, field_user_project_image)

API Tools and Resources

- POSTMan / Newman
 - <https://www.getpostman.com/>
- JSONView
 - <http://jsonview.com/>
- apiary
 - <https://apiary.io/>
- API Design, The Musical
 - David Diers

Unless you are creating something that will only ever be consumed by your own personal applications (and you are willing to accept your teeth gnashing in the future as you curse the house of your former self) treat your API as a contract. Once it's in the wild, it should not change.

Of course, you will obviously have to change some things, and that's where versioning comes in. Views doesn't make this super easy, but it's manageable.

When you are creating your API, you want to expose “interface” type names (name, project_name, user_image) not “implementation” names (field_user_name, field_project_name_2, field_user_project_image)

Demo Time!



We're Hiring!

We're currently looking to grow our great distributed team, with openings for a Senior technical project manager, a Business development manager and a Junior Drupal frontend developer



<http://fourkitchens.com/careers>

We're currently looking to grow our great distributed team, with openings for a Senior technical project manager, a Business development manager and a Junior Drupal frontend developer



Thank you!



All content in this presentation, except where noted otherwise, is [Creative Commons Attribution-ShareAlike 3.0 licensed](#) and copyright Four Kitchens, LLC.