

Information Architecture to Drupal Architecture

Outline

- Information Architecture (IA)
 - Why
 - What
- Mapping IA to Drupal – organizing content
- Working within a wider project process

Information Architecture to Drupal Architecture

- How do I build a site that:
 - Meets my requirements
 - Meets my users' requirements
 - Makes it easy to find things
 - Makes it easy to do things
 - Kicks ass



Information Architecture

- Analysis of:
 - Business requirements
 - Audiences and their needs (stories)
 - What content is (or will be) available on the site?
 - What are the functional requirements?
- Related disciplines:
 - User centered design
 - User experience (UX), usability

Information Architecture

- Values:
 - Make mistakes early and often
 - Iteration towards concrete plans
 - Users are important
 - Business needs are important
 - Good organization of content
 - Usable interfaces

Information Architecture

- Outcomes:
 - High level structure – site map
 - Content matrix/listing and examples
 - Cross cutting content structures – taxonomy, metadata
 - Wireframes – bringing together page elements into a cohesive user experience
- Some examples...

So now what?

- We know what the user experience should be for our site
- We have some nice greyscale wireframes...



- How do we start mapping that to Drupal?

Process

Easy!

- Step 1: Map
- Step 2: Reduce
- Step 3: Prototype & build

I lied.

- (Step -1): Know what Drupal can do OOTB and review often so that you can keep your IA in line with your budget/time
- (Step 0): Keep IA focused on the stuff most critical for launch. Don't make a YouTubeFlickrFacebookEnterpriseSite...
- Step 1: Map
- Step 2: Reduce
- Step 3: Prototype & build

Starting to break it down

A strategy for content

- Mapping the IA to general functionality:
 - 1) Managed navigation
 - 2) User driven categorization and metadata
 - 3) Displaying enhanced listings
 - 4) Managing page layout

1) Managed navigation

- Stuff that you, or a site admin would create and would be somewhat 'built into' the site, or change relatively infrequently
- Tools:
 - Menus
 - Content-types
 - Taxonomy

Might look simple, but look at link destinations and understand what this affects (e.g. URL path structures and content listings)

2) User driven categorization and metadata

- Generally evolves and changes at a rapid rate, larger number of elements to consider
- Tools:
 - Taxonomy
 - CCK
 - Node Reference
 - Organic Groups

3) Displaying enhanced listings

- Ways for users to find content using (especially) user driven categorization and metadata
- Tools:
 - Views
 - Custom code

Content sorting rules are important.

Finding the 'hidden fields'.

4) Managing page layout

- Ways of managing where on the page different navigational elements, listings and functional elements should go
- Tools:
 - Blocks/regions
 - Panels
 - Theming

”Everything else”

- The above is a strategy for mapping content structure needs to Drupal tools
- A similar process works for other requirements and more interactive pages:
 - Search
 - Authenticated user actions (login, profiles, content creation, voting etc)
 -

Uhhhhhhh

- That looks hard
- Think outside the box
- Check existing modules
- Abstract it – focus on the most reusable parts (talk to other people)
- "Phase 2" it – prioritize, prioritize, prioritize
- Review your IA better in the first place
 - 'Serious' and critical custom code specs & estimates must be done early on. No surprises.

Hmmmmmm

- I can see 42 ways of doing that
- Which is the best way?
 - Simplicity
 - In line with the rest of the site
 - Upgradability
 - Flexibility – ability to change with future requirements (some examples...)

Where to go from here?

- So I think I have some idea how this all comes together.
 - What should I do next?
- Document the plan
- Annotation
- Prototyping

In the project context

- Waterfall to agile
- Visual design
- Theming
- Module development
- Constraints:
 - Budget
 - Timeline
 - Functionality
 - Kitteh



Risks

- Clients with no clear business requirements
- Client delays around content etc
- Expectations (around client time involvement, prototyping)
- No technical review during IA process
- Incomplete/insufficient IA

Why IAs need to learn Drupal

- Projects don't exist in a vacuum
- The perfect user experience is no use if the project never launches
- Drupal has many good IA components that can be reused
- Drupal needs input from IAs to improve theming, functionality and usability/UX

Why Drupal geeks need to learn IA

- It helps your project succeed in the broadest sense
 - Not as simple as just 'doing what they client says'
- It helps improve Drupal
- You can help smaller projects succeed without needing a whole team

Thank you!

- Any questions?

Owen Barton

(AKA Grugnog)

<http://drupal.org/user/19668/contact>

CivicActions
Empowered