Umbraco US Festival 2024

Reduce, Reuse, Recycle:

Improving the Architecture of an Existing Site

Heather Floyd

Mary Shelley, 1831

painting by
Samuel John Stump (1778-1863)

Attack of the Frankensite

Hox Moke
FRANKENSTEIN

How is a Frankensite created?

(Unintentionally)

1. Starting with a design-specific architecture
2. Developer turn-over combined with a constant “fire-drill” working pace.
3. Problems fixed with band-aids

Problems with a Frankensite

Tedious for Editors

1. Extra properties not being used
2. Unclear how to update things on the visible website
3. Obsolete “required” properties or duplicate editing efforts

Problems with a Frankensite

Confusing for Developers

1. Extra unused “stuff” everywhere (more doctypes, code files, etc.)
2. Unclear what things are actively needed – fear of messing up/deleting something still in use
3. Hard to fix bugs because of byzantine sprawling codebase

Problems with a Frankensite

Inefficient

1. Front-end running slowly
2. Server resources used for the storage of unneeded files, and compiling unneeded code
3. Human time wasted wading through the muck

General Feeling of DOOM

Getting Approval for a Cleanup Project

Challenges

1. Time and dedicated resources are needed
2. Needs to be done outside of the daily content grind
3. Might not have an obvious visible difference at the end

Use an Upgrade!

1. It’s easier to justify that it’s a big project
2. Justification to set up a separate, new environment for the project
3. And… Maybe a visual design refresh?

Sell it

1. More stability
2. Save future dev & editor time
3. Easier, faster functional updates

Planning the Project

Planning the Project

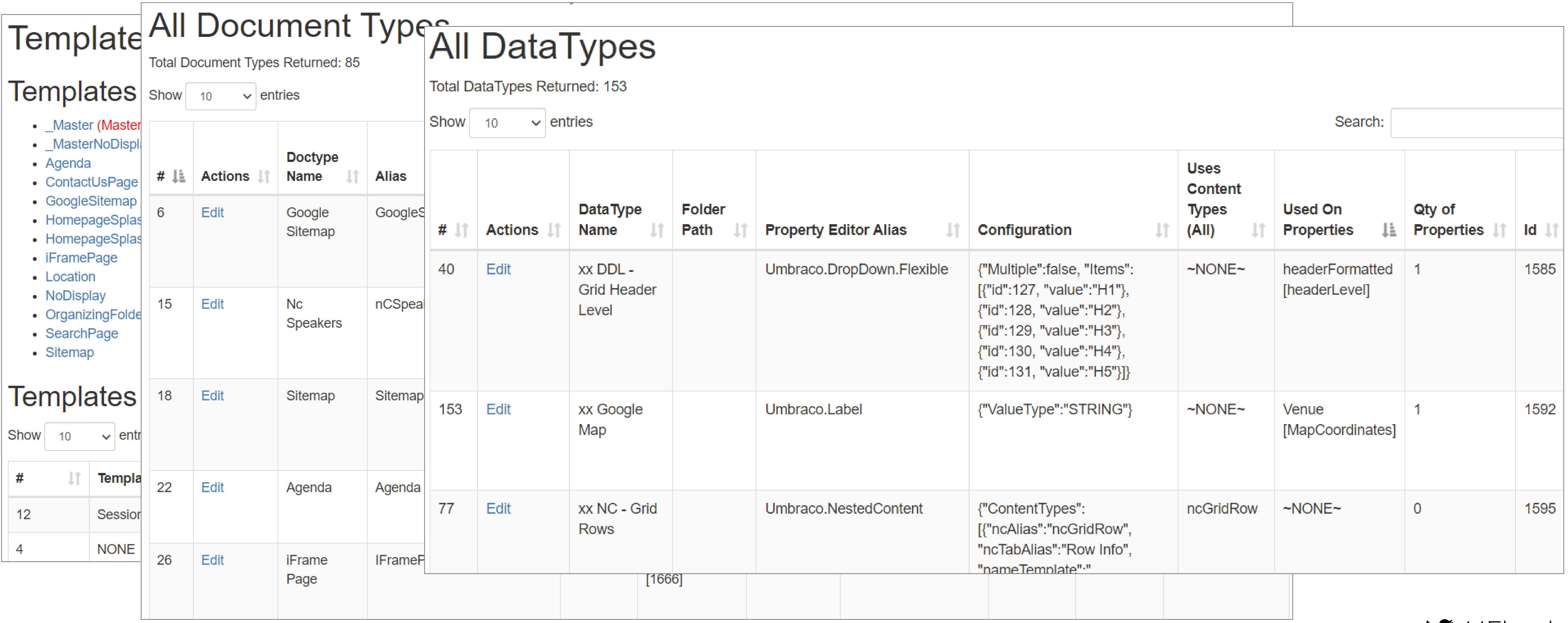
1. Estimate way more time than you think you'll need.
2. New design? Take care of that completely first before proceeding.
3. Plan for a development pause for the live site.
4. Plan for a "Content freeze"
5. Upgrade the database copy (if doing a major version upgrade), then copy over the code files only as needed

Cleaning Up

Cleaning Up

Look for things not in use

(Dragonfly Site Auditor)



Template

Templates

- _Master (Master
- _MasterNoDispl
- Agenda
- ContactUsPage
- GoogleSitemap
- HomepageSplas
- HomepageSplas
- iFramePage
- Location
- NoDisplay
- OrganizingFolde
- SearchPage
- Sitemap

Templates

Show 10 entr

#	Templa
12	Session
4	NONE

All Document Types

Total Document Types Returned: 85

Show 10 entries

#	Actions	Doctype Name	Alias
6	Edit	Google Sitemap	GoogleS
15	Edit	Nc Speakers	nCSpea
18	Edit	Sitemap	Sitemap
22	Edit	Agenda	Agenda
26	Edit	iFrame Page	IFrameP

[1666]

All DataTypes

Total DataTypes Returned: 153

Show 10 entries

Search:

#	Actions	DataType Name	Folder Path	Property Editor Alias	Configuration	Uses Content Types (All)	Used On Properties	Qty of Properties	Id
40	Edit	xx DDL - Grid Header Level		Umbraco.DropDown.Flexible	{"Multiple":false, "Items": [{"id":127, "value":"H1"}, {"id":128, "value":"H2"}, {"id":129, "value":"H3"}, {"id":130, "value":"H4"}, {"id":131, "value":"H5"}]}	~NONE~	headerFormatted [headerLevel]	1	1585
153	Edit	xx Google Map		Umbraco.Label	{"ValueType":"STRING"}	~NONE~	Venue [MapCoordinates]	1	1592
77	Edit	xx NC - Grid Rows		Umbraco.NestedContent	{"ContentTypes": [{"ncAlias":"ncGridRow", "ncTabAlias":"Row Info", "nameTemplate":"	ncGridRow	~NONE~	0	1595

Cleaning Up

Consolidate your Data Types

Default naming structure for a Data Type:

Content Type Name – Property Name – Property Editor Name

"Homepage - Top Right Blurb – Textbox"

Preferred naming structure for a Data Type:

Property Editor Name – Something about its configuration

"Media Picker – Single Image"

"MNTP – Products"

Cleaning Up

Look for other areas for improvement

1. DRY (Don’t repeat yourself)
2. Numbered Properties => single “multi” property
3. Scrutinize anything named without inherent meaning (ex: “Top-right content”)
4. Umbraco Deploy Data migrations tool

Cleaning Up

Identify and handle “legacy” properties

1. Label properties “LEGACY”
2. Place legacy properties near the “new” property, or create a “LEGACY” group for them
3. Provide description text to guide editors to new properties
4. Use smart fall-backs in your updated Views

Cleaning Up

Rethink front-end organization

1. Perhaps utilize SASS or LESS
2. Consider your front-end framework

Cleaning Up

Check/update all the Templates

1. Go section-by-section, methodically
2. Move complex logical code out of primary view files
3. Shift errant CSS/JS into your centralized system
4. Use Resharper’s “Find Code Issues”

Cleaning Up

QA all the things

1. Tip: Install Our.FulltextSearch and let it open all the pages via HTTP. Then check the Log for errors.
2. Have humans click around both old and new pages looking for any visual issues as well.

Future-proofing Your Architecture

Heather Floyd

HFloyd

Future-proofing

1. Make sure properties have MEANING
2. Use Blocks for structured free-form content
3. Separation of content and design
4. Add boilerplate logging to Macro partials and any partial views you might want to sunset.

Final Thoughts

Talk Resources:

https://HeatherFloyd.com/us-festival-2024