Some things about coding and stuff.

Tom Morris - @mozzy16

Disclaimer

These are all opinions that have been formed over the years of being a developer. It is by no means a ‘you should be doing this guide’, and more like a ‘these are observations that tend to make things easier’ type of approach.

Project structure

Typical solution

- Umbraco website application, e.g. Project.Web.UI

- Class library for code, e.g. Project.Web.Core - controllers, models, helpers

- Tests folder, add tests relating to each project

- Global folder, add editorconfig, gitignore, readme

Will cover most scenarios, you might want a data layer, microservices also a good approach.

Organisation

- Create folders relating to type, e.g. SurfaceController => /Controllers/Surface/MySurfaceController.cs

- Follow patterns and be consistent, help know where to look

- Split up your page into components, tie in with Nested/Stacked Content

- Split up other sections into partials, e.g. footer, header, meta.

- Client folder for FE stuff, build into static folder

Umbraco

Editing experience

- If something is likely to be edited, make it content editable

- Use descriptive names for properties, prefix if you want, add notes below, or use uEditorNotes

- Add appropriate icons, it does help and makes it look much better

- Use folders for doc types, data types, organise into things like pages, components, settings

Models

- Use strongly typed models, it's easy with newer Umbraco, get intellisense, quick demo of ModelsBuilder

- Can extend with your own properties if needed, less logic in views

- Use compositions for properties that are likely to be used elsewhere, easy to extend and share with other doc types

- You get interfaces when using compositions, which will help with your views.

Querying

- If you are querying content, don't do it again and again, create a variable, do upfront.

- Be careful querying content, as your site grows it can slow things down if done inefficiently, e.g. descendants. try XPATH

- Benchmark using MiniProfiler, included with Umbraco

Coding styles

Part 1

- Don't handle posts in your view, hard to read and limited options, create a controller

- App_Code is not a good place for lots of code, move to separate project

- Do document, add (useful) comments in your code. it can definitely help break things up and save the day for others

- Be descriptive with naming, say what things do. assign variables based on what they represent.

- Enforce conventions with editorconfig, linting tools

Part 2

- If there is code that is stretching the page a little, sit back and think how to break up

- If there is code you're doing quite a few times, move to extension method / helper / service.

- If you've got lots of query strings on your action methods, you can create a simple POCO

- Useful to create objects relating to configuration, rather than calling AppSettings each time

- Make use of form binding, i.e. don't treat a bool as a string

Scale

Dependencies

- If you need scheduled tasks, look at using Hangfire => persistant, proper times, not closely tied to Umbraco

- Or make use Azure functions, live outside of your web application, doesn't consume resources and cheap to run

- Make use of external services, e.g. blob storage, logging. again, not closely tied to Umbraco

- If you are using custom DB tables, PetaPoco is supported with Umbraco and you can make use of migrations (https://gist.github.com/sitereactor/5742799)

Build + deploy

- Have a build process that automates tasks, e.g. don't check in assets

- Run tests, code checks as part of the build, e.g. SonarQube

- Have a deployment process that allows rollback, can handle configuration and is easy to extend

- Use hosted services, e.g. Octopus Deploy, Team City, AppVeyor, VSTS

- Update Umbraco little and often, stops you from getting too far behind, hopefully makes it easier, get new updates. just probably don't do it straight away.

Self development

Help yourself

- Sit back from time to time and look at your problem

- Talk to others about how things work, quite often solve it as you speak

- Make use of PRs and tooling to support code reviews

- Attend meetups, make use of community for support

- Have fun! Do things that interest you.