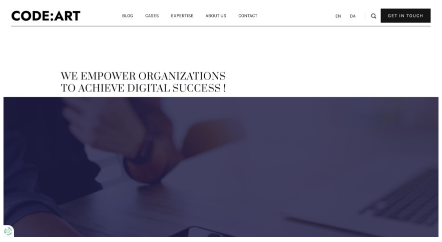
ALLAN THRAEN | 5 months ago | PDF |

Website Improvements Life as a Coder Freelancing Optimizely (Episerver) CMS

LIVE ON OPTIMIZELY CMS 12 AND .NET 8



"Better late than never" is a fitting saying here. We've finally gotten around to updating the CodeArt.dk website to Optimizely CMS 12 and .NET 8!

For the last couple of years - ever since CMS 12 came out, basically - I've been busy helping several of my customers upgrade/migrate/replatform their websites - and a many are still to come.
Usually, upgrading an Optimizely/Episerver site between minor versions is quite straightforward; Upgrade nuget packages and dependencies, potentially fix a few places and do a full retest. And - to be honest, from EPiServer CMS 7 - 11, the major versions haven't been cause for much trouble either. At least not if the site wasn't based on outdated technology such as WebForms, XForms or the like.

Typically a couple of days of developers going through the list of breaking changes and checking/rewriting bits and pieces of the code - and then a full retest and launch.
But as you probably know, the move from CMS 11 to CMS 12 is quite a different beast. Mainly because it's also a move from the .NET Framework to .NET Core - which potentially means a lot (!) of things has to change.

I usually advice my clients to consider if this would be a good time to do a re-implementation of the entire web experience - rethink, restructure, freshen up - as many like to do every few years. That works for some.
Others "just" want an upgrade - but even though some of the automated code-update scripts can be a decent help to get going, they won't take you all the way and they'll likely leave some tech debt behind.

One approach that I've used successfully on clients that 'just' wants an upgrade is to do a parallel approach. Use the code upgrade tools to upgrade as much of their codebase as possible, and in parallel start op a new well-structured .NET core solution suited to their needs and then copy in the upgraded code manually step-by-step. While manually migrating code, make sure to only migrate code that actually is used - and not old stuff lying around the codebase because no-one dared deleting it. Even more ideal if you have a chance to clean-up the existing codebase before starting the migration project - that way there is less code to upgrade!

Our upgrade story

We have a saying in danish that loosely translates to "the shoemakers children wear worn down shoes", meaning that when you do something professionally for others, you yourself will be last in line (is there a corresponding english proverb? Let me know in the comments). When it comes to websites I often find that to be the case as well. Paying customers tends to take precedence over our own website - which of course has slowed down the process of upgrading our site quite a bit - at the same time, the site was extremly customized as I had used it for as a demo/playground for many smaller projects over the year - many of which it doesn't make sense to upgrade (or they haven't been upgraded quite yet).
The actual upgrade of the main website codebase only took a couple of days - and was done a long time ago. Rewriting all the needed add-ons and extensions took a lot longer - and the site is still far from done. At the same time we decided to give the UI a little freshen up - as well as logo and the general visual impression. All look & feel and frontend coding is of course done by our very own Kseniia Zar.
We started development on .NET 6, but switched to .NET 8 as soon as it was officially supported.

For those interested I can mention the extensions we've used so far:

- EPiServer Search & Navigation (but already considering replacing it with Content Graph)
- Advanced Reviews
- Our own Property Inheritance module
- The CookieInformation integration
- DeveloperTools
- Geta's NotFound handler
- Geta's ContentType icons
- Geta's Sitemap
- Powerslice
- ImageSharp
- And a bunch of custom add-ons not yet released (like Excel content report generation, Blazor Content Audit, Blazor DDS Explorer and so on)

Should you upgrade

If you are running EPiServer CMS <12, then you should be at least planning for an upgrade. Sure, "If it works don't fix it" is of course a methology to follow - but I always recommend people constantly evolve and improve their online experiences - otherwise you'll eventually be stuck behind a mountain of technical debt and see yourself surpassed by your competitors. And all new features coming our are of course targeting the latest platform - while CMS 11 only gets the most essential bug-fixes og security patches.
I realize it might be a hard sell internally. An upgrade can be costly and the upgrade in itself won't give you a lot of new desirable features. Significant performance improvements yes, better developer experience yes, nicer editorial experience yes - but those can be hard-to-sell features internally. However, I still strongly recommend that you upgrade - so you constantly can keep up to date with the latest features and add-ons - and over time evolve your web experience to be even better than it is today.

And - since it's a costly operation already, why not take the opportunity to re-implement significant parts of your architecture to be more streamlined with your business needs? And easier to maintain? Provide better online service? Better editorial experience? Better developer experience? It's worth considering.

Managing an upgrade project

So far, I've seen "upgrade" projects ranging from 3 days to 1.5 years. So naturally, there is a big risk in it. And while the upgrade work is going on, you typically can't pause the 'regular' web site development to

your current platform. Bugs still needs fixing and business requirements needs implementing.
I've partnered up with some really competent colleagues at Epinova Sweden and we've come up with a model that works very well:
A small dedicated external expert team comes in and does the upgrade in parallel while the existing dev team maintains the old solution. To minimize risk and unfortunate surprises we typically does it in 3 steps:

1. Pilot project: Small initial project of around 2-300h at a fixed price where the upgrade is started and as many problem/risk areas as possible are identified - and most architectural decisions that needs to make are identified. Often, by then end of this pilot project, there will be a prototype of a running solution on CMS 12, and a list of identified remaining task sets the foundation for some thorough estimates on the rest of the upgrade.
2. After discussing all major decisions with the client and the existing development team and agreeing on the limitations of the upgrade, the project is continued towards completion.
3. In the final phase of the project the developer team maintaining the old site starts to get introduced to the new site and assist in re-implementing any new features that has been added to the old site since the upgrade project started.

Feel free to reach out for a talk to see if this approach could be right for your site.

Website Improvements | Life as a Coder | Freelancing | Optimizely (Episerver) | CMS

RECENT POSTS

CodeArt ApS
Teknikerbyen 5, 2830 Virum, Denmark
Email: info@codeart.dk
Phone: +45 26 13 66 96
CVR: 39680688