← Blog posts

Content Provider or Content Replication

OPTIMIZELY (EPISERVER), ADDON DEVELOPMENT, INTEGRATIONS, CMS

Allan Thraen | 3 Years Ago | PDF |

When integrating external content into Episerver, a classic dilemma is whether you should replicate it in, or setup a content provider to pull it in real-time. As part of the Digizuite Integration I have once again given some thought to the dilemma - and here are some pro's and cons.

Ok, fair warning - I've had a secret passion for content providers ever since they were called 'Page Providers' and was introduced a long, long time ago. The entire concept of writing a provider class and magically expose anything as content (in any way you want), while maintaining a single source and specific ownership is just plain awesome. But it does come with some serious drawbacks you need to deal with as well.

On the other hand, replicating content (typically done with a scheduled job (pull) but could potentially also be event driven (push)) is fairly easy to do - as long as you don't need it real-time, only need to read it, it's a limited volume - and you don't mind having duplicate stuff lying around.

Let's consider our options in more detail (+ are positive aspects, - are negative, duh):

Content Provider

+ Elegant. *Ok, I'll admit that's a matter of taste. But keep in mind that Episerver itself stores content through a content provider - so it kind of fits with the flow.*

+ Single Source

+ CRUD. *Assuming you implement the required methods*

+ You have full control of pretty much all aspects (including caching, searching, browsing behavior)

+ Scalable. *But as always you have to consider how you structure the content it outputs*

+ Search. *If you implement it correctly*

+ Secure. *If you implement it correctly*

+ Supports Upload of assets (through blob provider)

- Typically requires a lot of pretty complex code

- Expects content+structure+functionality like that in Episerver. *It's not uncommon to have to find workarounds for ID-mapping, versioning, draft workflows and so on.*

- Resilience is tricky. *You always depend on the external source working and being accessible - unless you find a way to work around that.*

- Caching is tricky. *But essential. You need to ensure a way to update your cache based on external changes, yet still have an active cache for performance reasons.*

- Usually doesn't stand alone. *Meaning you often have to consider blob providers, search providers and so on.*

- Mapping is tricky. *There is a mapping service to help you map external ID's to Episerver ID's+Url-segments+Guids - but it has it's own set of problems.*

Replication

+ Simple. O*ften can be done with a single scheduled job*

+ Relatively easy to build and maintain. *Fetch external content (or ideally changed external content), update and insert locally (as Episerver Content / structure) based on predefined configuration / hardcode. Done.*

+ Resilient. *As all content is just plain episerver content, we don't add new dependencies that has to be running, except for when we synchronize.*

- Only one-way. *Sure, you can allow your editors to change it in Episerver, but typically those changes won't push through to the original source. For that to happen you'll have to code a specific event handler that does that.*

- No single point of truth. *Assuming you are not doing 2-way synchronisation you risk your Episerver replicated copies is edited to be different from your external source.*

- Typically not real-time.

- Potentially unsecure. *If you only set access rights on synchronization - and that is not real time, access rights changes in the source might not go into immediate effect on Episerver.*

- Scalability can be an issue. *Assuming that your external repository scales better than Episerver for storing and working with large amounts of the specific content structure you use.*

- Relying on Episerver Search. *Which is fine if you use Find - but not great if don't.*

- Fast synchronization can be tricky. *Unless you can store Episerver ID's in your source data or maintain a mapping table.*

So, as always there are pro's and con's to each approach and it really depends on the specific use case what you should do. In the current project where I'm assisting Digizuite with a deep integration to Episerver for their DAM, it's been essential to make it as native to Episerver as possible - including real-time updates, full read-write capabilities, good search and scaling - so we've gone with a Content Provider approach - but am doing a lot of work to improve the parts where content providers are challenging - including adding aggresive caching, additional resilience work and many supporting providers.

I'd love to hear your take on this - so feel free to comment with your war-stories of cases where an external source delivers Episerver content.

OPTIMIZELY (EPISERVER) ADDON DEVELOPMENT INTEGRATIONS CMS

Post Comments 0

Digizuite DAM for Episerver
Digizuite: Keeping developers in mind when building the addon
Gist Content Provider
Building your own Page Provider: Northwind
Mapped Page Provider
Codemania: Geeky PageProviders