ALLAN THRAEN | 5 years ago | PDF |

Tips and Tricks Integrations Optimizely (Episerver) CMS

CONTENT PROVIDERS AND FLAT CONTENT

A classic challenge in many CMS - and also in Episerver - has always been what do you do with large amounts of non-hierarchical/flat content? There has been many workarounds along the way and I was just on my way to make yet another when I discovered a well hidden secret deep in the belly of Episervers UI: The Asset widget (that holds blocks and media items) does in fact have infinite scrolling - which in turn can support incredibly large flat structures!

I guess the problem often has shown it's way in the shape of a lot of blog posts, news articles - or (as in my case) a lot of images without a logical hierarchy. The problem: you tend to run into problems when you have a lot of content items under the same node. Most of the problems, though are very basic and connected to the Episerver UI - when you expand a node in the conten tree it'll try to load all children at once - and if there are a lot (10.000+) that can take a while and be a bit heavy on the UI.

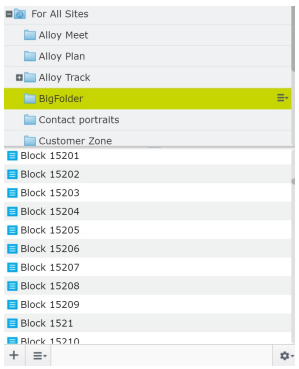
The classical solution in the epi-world has been to inject a hierarchy of containers that made sense. Like I've done and demonstrated in the blog post about building an automatic blog hierarchy. However, in my current case of a lot of images coming from a new content provider, I can't always force a sensible structure where none is to be found.

Obviously, the UI issue has been solved for a long time in parts of the Episerver UI - like the Commerce UI that has catalogs with any number of items in them. But the CMS sister-system has been struggling. Grzegorz Wiecheć made an impressive approach to solving it using magical dojo skills that must have taken at least 5 years at Hogwarts to learn, with the Content ChildrenGrid View - but I haven't seen that available in the nuget feed yet, so I can't use that.

A side node here is that you can of course study what other systems do - several epi-competitors uses the concept of 'buckets' - nodes that you can pour flat content into and that then has invented other ways of making it available in the UI than a classic tree structure. *But in some cases I suspect that they simply just hide the content in the tree below bucket nodes so it's only available through search or other alternative gridviews - something that is fairly easy to implement in Episerver as well.*

Anyway, it turns out that the lovely Asset widget in Episervers edit mode is in fact born with infinite scrolling for in the Content list (bottom part) which to a certain degree does solve the UI issue - as long as the flat content is assets of sorts and not pages in the page tree.

When you look at the dojo behind the scenes, you'll see that it's based on the HierarchicalList widget which contains a searchbox widget, a foldertree widget and a contentlist widget. The contentlist widget in turn (through a longer inheritance chain is in fact a dgrid OnDemandList - which has infinite scrolling! Similarly, it calls a RestStore in the backend which supports a Range parameter. It might not sound like much but this is huge. And made me dance around the office in utter joy and amazement for a good 30 seconds :-)
This is indeed a hidden jewel! To verify the assumption I had from digging through the code I obviously wanted to test it out, so I made a scheduled job (my regular go-to for batch jobs in Episerver) that creates 30.000 flat Editorial blocks in a content folder on an Alloy site. Then I monitored the AJAX calls with chrome developer tools and sure enough - it does a lot of calls to the rest store ('/EPiServer/cms/Stores/contentstructure/?references=124&query=getchildren&allLanguages=true&typeIdentifiers=episerver.core.blockdata&sort(+name)') with a HTTP Header called 'range' indicating the range requested. And when I start scrolling the calls keep coming and the data is filling up.



Content Provider Trick

As cool as this was, I wasn't completely happy though - cause showing a lot of items still took too long in my Content Provider - and if you consider the flow it's not all that strange:

Typically when you implement a content provider you start out with the most important method - the one that is in fact required: LoadContent(...). That's the one that returns IContent and without it you can't provide any content, hence no content provider.

Secondly you might start to consider the content hierarchy and override LoadChildrenReferencesAndTypes() method. And all is fine.

So - when the Asset widget calls the rest store that in turns calls your content provider cause a node has been expanded, it'll first get the children references and types. All of them. But as long as you're fairly quick to provide them that's fine. It is after all only references and not the entire content. And by default that list gets cached as well. But then the trouble starts, cause the default behavior is then for it to call

"LoadContent" on every single one of the content items requested in order to fetch the name, thumbnail and similar for displaying it. And that's a problem if you (as many content providers do) rely on an individual http call to fetch the content item in question. That suddenly becomes a lot of http calls when getting details for 38 items (or however many the list wants to show) simultaneously.

The trick is to **override LoadContents(IList<ContentReference> contentReferences, ILanguageSelector selector)** - this way you can perhaps make only one http call fetching the requested list of content references!

Why does this only take us 90% of the way

Well - sadly, the Asset widget isn't the only place you can browse assets. There's for instance also asset picker dialogs various places (which does not have infinite scrolling) and a couple of places in admin mode it's possible to browse the entire content tree - in an old fashioned tree view that certainly also doesn't deal with the problem of flat content. My approach for admin mode is so far to simply detect if that's the context calling my provider and then not return individual assets. I'm still working on a strategy for the dialogs, but I'm fearing it might involve sacrifising a lamb on a hidden dojo-altar in a dark corner of the world :-)

Update 2018-10-02: Grzegorz Wiecheć has done yet another of his awesome posts, https://gregwiechec.com/2018/10/image-property-with-media-component/, with an Image Asset Picker dialog that uses the full media browser - including infinite scrolling! One more step of the way there.

Tips and Tricks | Integrations | Optimizely (Episerver) | CMS

RECENT POSTS

Automatic Blog Hierarchy
Content ChildrenGrid View
Gist Content Provider
Image Picker with Media Picker

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