← Blog posts

Exploring the Episerver Nuget Feed

OPTIMIZELY (EPISERVER), BIG DATA, VISION DEMOS & PROTOTYPES, INFORMATION RETRIEVAL

Allan Thraen | 2 Years Ago | PDF |

The best thing about Episerver is the community and all the great contributions coming from it. Many of them make it into packages on the Episerver nuget feed - along side Episervers own packages. I have for a long time worked on building tools to explore and visualize this more - and now I'm finally ready to one-by-one share some of the tools coming out of it.

Episervers own official Nuget Browser (https://nuget.episerver.com/) is pretty cool and very useful as a tool to find nuget packages and see their different versions. And I have for a long time been a fan of David Knipes Nuget Feed Explorer where he has indexed all the meta-data about the nuget packages and made it easy to browse them (and even see flow charts of their versions).

But for a long time I've still felt like I often ran into questions that the nuget packages could answer - but not with the tools currently available.

First of all, I'd like to be able to list and locate nuget packages - not just based on their official metadata, but on a whole lot more. For example, what do they contain? Which types are defined in their code? what do they inherit from? Where is a certain interface defined? Are there any packages with a SQL Script that touches on a certain table? Which packages contain code that inherits from class X and so on.

Secondly, I want to be able to play with the data and see charts and trends across multiple packages. How active have the community been in releasing new packages over time? What is the top 10 of EMVPs that has produced the most popular packages? Can I see a chart indicating which packages are 'hot' comparing their popularity and released versions?

Lastly, I would like to learn about changes in the feed as close to real-time as possible. Often, people write blog posts about what they have put out there, but it would be nice to be able to get alerted when a new version of your favorite package is out, or when an author your follow has created a completely new package. Or what about instantly finding out if your own package has reached the popularity top 20?

On top of all of this I love exploring datasets, and this is a really interesting one to work with. I expect this will just be the first blog post of several talking about this project.

Indexing it all

I started off setting up an ElasticSearch index to hold all the data and be able to query it as well as easily aggregate data.

I wrote a console application that every night runs through the nuget feed, indexes the metadata for each package - but also downloads the package.

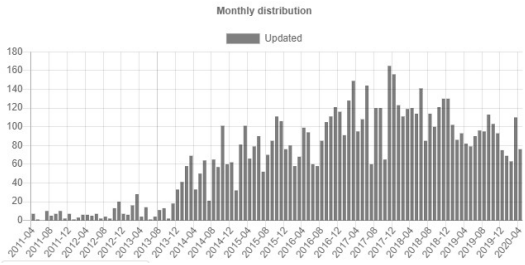
A nuget package is essentially a zip file, so my program then unzips it and goes through every file contained in it, indexing every file.

That is, indexing the meta-data of the file - and if possible the contents - like for .js, .css, .sql, .cshtml, ... files.

If I stumble across an assembly, my code will do a more thorough analysis, extracting all types in the dll and a bunch of meta-data for each type. Stuff like:

- What does it inherit from
- Which methods does it have
- Which properties
- Which events
- Is it public
- Is it abstract
- ...

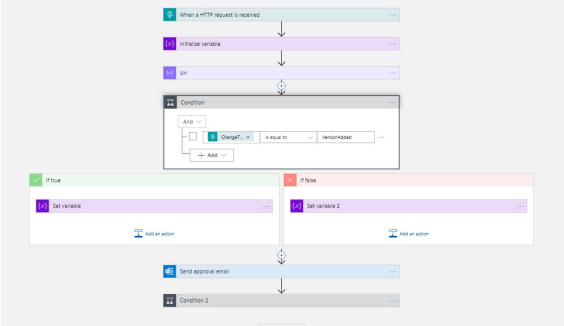
Now, I have all this data and I'm working on ways to visualize it and use it. This is still very much work in progress, but here's for example a basic chart view of packages released over time on the feed:



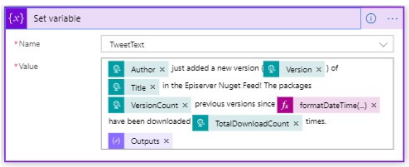
Another example is that recently (for another project) I needed to know how common it was to use Episervers GetChildrenQuery class, and a quick search indicated that it was done 45 places in 6 packages. And which packages they were.

But you guys will have to wait a bit on my UI skills before all of this goodness is available to you :-(

One small thing I did start with, is that I now daily run a job that identifies interesting changes - like new packages or new versions in the feed. When they are detected, it calls a webhook I configured in a nifty little Azure Logic App, which then starts a workflow, composing a tweet about it. Well, actually 2 tweets - one if it's a new version, and another if it's a new package all together. It looks like this:



You can see the actual tweet being composed here:



It then sends an email to me asking me to confirm that I want to tweet that (I might remove this step later when I'm sure it's working as intended) and if I click accept in that mail, it will instantly spam the twitterverse with it :-)

Stay tuned for more updates on this - and follow https://twitter.com/athraen to get the latest nuget updates straight in your twitter feed! I'd love to hear your ideas, usecases and suggestions for what I can do with this - drop a comment below or reach out directly!

OPTIMIZELY (EPISERVER) | BIG DATA | VISION DEMOS & PROTOTYPES | INFORMATION RETRIEVAL

Post Comments 0