

[← Blog posts](#)

Powerslice: Identify unused blocks

ELASTICSEARCH, TIPS AND TRICKS, OPTIMIZEZY (EPISERVER)



Allan Thraen | 2 Years Ago | PDF |



Powerslice might have a few years on it's back, but it's still a great editorial tool, when you are working with large amounts of content and have access to Episerver Find. Here's an example of a recent slice I made that let's editors easily identify unused blocks.

Powerslice is a nice widget that gives the editors an alternative to the classic tree navigation when working with content.

In the backend it is powered by Episerver Find, and as a developer it's pretty easy to extend with your own views of the content.

Typical extension points are:

1. Slices with different filters for content. For example a slice that lists all ArticlePages that are not 'Shortcut to another page'. Or a list of blocks that was created by the current editor - or maybe a list of all the pages that does not have proper SEO meta-data set?
2. Custom sort orders for the slices. Maybe you want to sort by content type? Or what about showing the oldest published content (that probably needs a rewrite)? Sometimes it could also be interesting to see a list of blocks sorted by which are used in the most places (example below).
3. Finally you can also provide editors with a shortcut to create content of a specific type in a specific location. For example, when they are looking at a slice of all Blog Posts, you can let them create a new blog post in the correct location - straight from the widget.

A client of mine had a specific problem that I think many others can recognize. They have many blocks - placed all over the possible structure - and they need to clean them out a bit - evaluate the unused ones to see if they should be deleted or archived. So - here is one approach I made to this problem:

Initial challenge is that the usages of a block is not stored on the block itself, and as such it's not indexed in Find and cannot be filtered on. But luckily that's easy to overcome.

I began by creating an extension method for IContent that would essentially return the number of times a content was referenced / used in other content, following a reverse lookup in the SoftLink Repository.

Now, before you start yelling "But Allan, how dare you use the ServiceLocator.Current in a public blog post?" just know, that I tend to take a more pragmatic approach than a dogmatic approach - especially when I want to keep it in a pretty, little extensionmethod - you feel free to pick your own way.

Next, the trick is simply during initialization to configure an indexing convention in Find where you specify that the extension method is to be included when indexing objects of the type you want to add this value to.

Finally, it's just a matter of doing a basic Episerver Find filtering query in the right place in the slice using our new number.

For pure bonus, I also included a sample of an All-Blocks slice, that will allow you to sort based on how frequently the blocks are used - as that may also come in handy.

```

1 using EPIServer.Core;
2 using EPIServer.DataAbstraction;
3 using EPIServer.ServiceLocation;
4 using System;
5 using System.Collections.Generic;
6 using System.Linq;
7 using System.Web;
8
9 namespace Demo.Business.Powerslice
10 {
11     public static class BlockSliceExtensions
12     {
13         public static int UsageCount(this IContent cnt)
14         {
15             //Count usages
16             var _soft = ServiceLocator.Current.GetInstance<ContentSoftLinkRepository>();
17             var links = _soft.Load(cnt, ContentLink, true);
18             return links.Count();
19         }
20     }
21 }
22 }

```

BlockSliceExtensions.cs hosted with ❤ by GitHub

[view raw](#)

```

1 using EPIServer.Core;
2 using EPIServer.Find;
3 using EPIServer.ServiceLocation;
4 using EPIServer.Shell.ContentQuery;
5 using Powerslice;
6 using System;
7 using System.Collections.Generic;
8 using System.Linq;
9 using System.Web;
10
11 namespace Demo.Business.Powerslice
12 {
13     [ServiceConfiguration(typeof(IContentQuery)), ServiceConfiguration(typeof(IContentSlice))]
14     public class BlocksSlice : ContentSliceBase<BlockData>
15     {
16         public override IEnumerable<SortAction<BlockData>> SortActions => base.SortActions.Union(new SortAction<BlockData>[] {
17             new SortAction<BlockData>("Most Popular","popular", (a,b) => a.OrderByDescending(tlk => (tlk as IContent).UsageCount()))
18         });
19         public override int SortOrder => 20;
20
21         public override string Name
22         {
23             get { return "All Blocks"; }
24         }
25     }
26 }
27 }

```

BlocksSlice.cs hosted with ❤ by GitHub

[view raw](#)

```

1 using System;
2 using System.Linq;
3 using EPIServer.Core;
4 using EPIServer.Find.ClientConventions;
5 using EPIServer.Find.Framework;
6 using EPIServer.Framework;
7 using EPIServer.Framework.Initialization;
8 using Demo.Business.Powerslice;
9
10 namespace Demo.Business.Initialization
11 {
12     [InitializableModule]
13     [ModuleDependency(typeof(EPIServer.Web.InitializationModule))]
14     public class FindIndexingConventionsInit : InitializableModule
15     {
16         public void Initialize(InitializationEngine context)
17         {
18             //Ensure the UsageCount is indexed along with the block
19             SearchClient.Instance.Conventions.ForInstancesOf<BlockData>().IncludeField(x => (x as IContent).UsageCount());
20         }
21
22         public void Uninitialize(InitializationEngine context)
23         {
24             //Add uninitialization logic
25         }
26     }
27 }

```

FindIndexingConventionsInit.cs hosted with ❤ by GitHub

[view raw](#)

```

1 using EPIServer.Cms.Shell.Ui.Rest.ContentQuery;
2 using EPIServer.Core;
3 using EPIServer.Find;
4 using EPIServer.ServiceLocation;
5 using EPIServer.Shell.ContentQuery;
6 using Powerslice;
7 using System;
8 using System.Collections.Generic;
9 using System.Linq;
10 using System.Web;
11
12 namespace Demo.Business.Powerslice
13 {
14     [ServiceConfiguration(typeof(IContentQuery)), ServiceConfiguration(typeof(IContentSlice))]
15     public class UnusedBlocksSlice : ContentSliceBase<BlockData>
16     {
17         public override int SortOrder => 750;
18
19         protected override ITypeSearch<BlockData> Filter(ITypeSearch<BlockData> searchRequest, ContentQueryParameters parameters)
20         {
21             return searchRequest.Filter(f => (f as IContent).UsageCount().Match(0));
22         }
23
24         public override string Name
25         {
26             get { return "Unused Blocks"; }
27         }
28     }
29 }
30 }
31 }

```

Post Comments 0

4 Comments CodeArt.dk Disqus' Privacy Policy

Login

Favorite Tweet Share

Sort by Best

Join the discussion...

LOG IN WITH OR SIGN UP WITH DISQUS

teachingbirds · 2 years ago
Hi! I just implemented your examples in powerslice for a customer website. It would however be great if they could also remove unused blocks directly from the powerslice gadget. Now they have to look it up in the assets pane too. Do you know of a way to do this?

Allan Thraen Mod · teachingbirds · 2 years ago
Funny you should ask. I'm currently working on extending the developer console to do just that (https://www.codeart.dk/blog/... It can't be done from the Powerslice gadget as the gadget only supports viewing lists of content and creating new - but no other actions.

teachingbirds · Allan Thraen · 2 years ago
Looks nice, but perhaps a tad bit too hard to explain to our editors how to use :) I will probably try to put a link in the settings tab on the block, to remove.

Allan Thraen Mod · teachingbirds · 2 years ago
Yeah, the Console isn't targeting editors. A planned feature for Developer console is "Editor Macros" where a developer can save a commandline and then editors can execute it easily. But we're not there yet :)
The easiest solution for your current problem is probably a scheduled job that uses a similar code as the one in this blog post to identify and remove unused block posts.