← Blog posts

Optimizely (Episerver): Split folder structure for blocks and media

TIPS AND TRICKS, OPTIMIZELY (EPISERVER), CMS, .NET DEVELOPMENT

Allan Thraen | 7 Months Ago | PDF |

Since version 7 or 8 of Episerver (now Optimizely CMS), the shared Blocks and Media have been sharing the same folder structure. Some people see a benefit with the shared structure, and some absolutely hate it. Personally, I have gotten used to it - but I was recently asked if it's possible to split it up. Here's the hack I came up with.

First of all, fair warning: *This is very experimental and I'm not exactly sure of all the consequences of this hack, so if you plan to use it - do so at your own risk (but feel free to leave a comment below with the good, bad and ugly). It hasn't been tested in a multi-site scenario and most likely won't work ideal there (although the code should be pretty easy to adjust for that). Also, it doesn't migrate any existing blocks or media - so that should be done before hand.*

The basic approach I took was to create 2 folders in the root of the GlobalBlockFolder - one for Media and one for Blocks. Afterwards I basically just needed the Blocks and Assets widget to use the new folders as their root. They both use an IContentRepositoryDescriptor for that, so it was pretty straightforward to create a new descriptor for each (inheriting their original), overriding the root and then using an IConfigurableModule to intercept the original calls using Dependency Injection.

The example below is not very optimized - but it does seem to work. If the folders doesn't exist, it will create them - and then make sure the widgets use them.

```
using EPiServer.Cms.Shell.UI.UIDescriptors;
using EPiServer.Framework;
using EPiServer.Framework.Initialization;
using EPiServer.ServiceLocation;
using EPiServer.ServiceLocation.Compatibility;
using EPiServer.Shell;
using System;
using System.Linq;

namespace ExperimentSite.Widgets
{
    [InitializableModule]
    [ModuleDependency(typeof(EPiServer.Web.InitializationModule))]
    public class InterceptorInit : IConfigurableModule
    {
        public void ConfigureContainer(ServiceConfigurationContext context)
        {
            //Override / remove
            context.Services.Intercept<IContentRepositoryDescriptor>(
                (locator, def) => (def is BlockRepositoryDescriptor) ?
                    locator.GetInstance<MyBlockRepositoryDescriptor>() :
                    (def is MediaRepositoryDescriptor) ? locator.GetInstance<MyMediaRepositoryDescriptor>() : def
```

```
                    );

        }

        public void Initialize(InitializationEngine context)
        {
            //Add initialization logic, this method is called once after CMS has been initialized
        }


        public void Uninitialize(InitializationEngine context)
        {
            //Add uninitialization logic
        }
    }
}
```

interceptorInit.cs hosted with ♥ by GitHub view raw

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using EPiServer;
using EPiServer.Cms.Shell.UI.CompositeViews.Internal;
using EPiServer.Cms.Shell.UI.UIDescriptors;
using EPiServer.Core;
using EPiServer.Framework.Localization;
using EPiServer.ServiceLocation;
using EPiServer.Shell;
using EPiServer.Web;


namespace ExperimentSite.Widgets
{
  public class MyBlockRepositoryDescriptor : BlockRepositoryDescriptor
    {
        private const string NAME = "Blocks";
        private readonly IContentRepository _repo;

        private ContentReference _root;

        public MyBlockRepositoryDescriptor(IContentRepository repo)
        {
            _repo = repo;

            //Get or create root
            ContentFolder rootfolder = _repo.GetChildren<ContentFolder>(ContentReference.GlobalBlockFolder).Where(f => f.Name == NAME).First(
            if (rootfolder == null)
            {
                rootfolder = _repo.GetDefault<ContentFolder>(ContentReference.GlobalBlockFolder);
                rootfolder.Name = NAME;
                _root = _repo.Save(rootfolder, EPiServer.DataAccess.SaveAction.Publish, EPiServer.Security.AccessLevel.NoAccess);
            }
            else _root = rootfolder.ContentLink;
        }

        public override IEnumerable<ContentReference> Roots
        {
            get
            {
                yield return _root;
            }
        }

    }


    public class MyMediaRepositoryDescriptor : MediaRepositoryDescriptor
    {
        private const string NAME = "Media";
        private readonly IContentRepository _repo;
        private ContentReference _root;

        public MyMediaRepositoryDescriptor(IContentRepository repo)
        {
            _repo = repo;

            //Get or create root
            ContentFolder rootfolder = _repo.GetChildren<ContentFolder>(ContentReference.GlobalBlockFolder).Where(f => f.Name == NAME).First(
            if (rootfolder == null)
            {
                rootfolder = _repo.GetDefault<ContentFolder>(ContentReference.GlobalBlockFolder);
                rootfolder.Name = NAME;
                _root = _repo.Save(rootfolder, EPiServer.DataAccess.SaveAction.Publish, EPiServer.Security.AccessLevel.NoAccess);
            }
            else _root = rootfolder.ContentLink;
        }
        public override IEnumerable<ContentReference> Roots
        {
            get
            {
                yield return _root;
            }
        }

    }
}
```

RepositoryDescriptors.cs hosted with ♥ by GitHub view raw

TIPS AND TRICKS OPTIMIZELY (EPISERVER) CMS .NET DEVELOPMENT

Post Comments 0