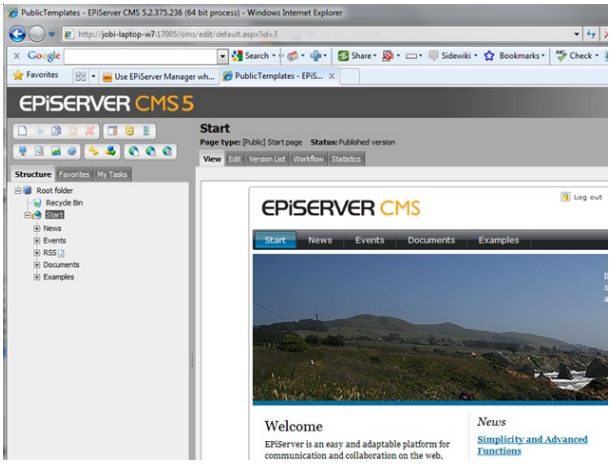


C#, EPISERVER

Good ol' Dynamic Properties

Allan Thraen | 1 Years Ago | PDF | 3 48 Shares

There was a time, when men were made of steel, ships made of wood, Episerver was spelled with a weird capitalization and the CMS had something called Dynamic Properties that was usually misused. They've been gone for a while, but I miss them, so here's yet another attempt at solving the property inheritance challenge.

To those of you, my dear readers, that have no clue what Dynamic Properties were all about, let me begin with enlightening you:

Once upon a time, there was a mythical feature, loved by some and hated by many called Dynamic Propeties. Dynamic Properties were properties that were not set on the content itself, but rather could be inherited across all content types, throughout the content hierachy. A good example would be a dynamic property called something like "SecondLevelMenuRoot" that would be used to know which child objects should be listed in the sidebar menu. Usually, a super-user editor that could find his/her way into the secret edit menu for dynamic properties would then set the property to the each of the first level items. That way, the property would inherit down and all leaf nodes would show the second level menu corresponding to their place. And the world was a better place.

Sadly, evil forced tended to abuse these great powers and often ended up making many, many dynamic properties and use them for stuff like a LogoImageLink or SearchButtonText that really should have been a site setting instead. Since Dynamic properties had to be resolved dynamically (hence the name) that tended to slow down the sites quite a bit.

But, surely we are smarter now and ready to once again unleash this power, right?

In any case, I needed some inheritance badly when building this blog. Why?

Well, here is a good use-case: On each blog post I have a sidebar content area. Usually I'll have the same blocks there on all blocks - but there'll probably one or two where I'll want something different placed there. I don't want to have to remember to put the same blocks in there for every single blog post. And I don't want it as 'Default' value upon creation, cause what happens when I decide to change the blocks?

I could perhaps make 1 giant Block to rule all blocks, that would contain another content area with the other blocks....But blocks-in-blocks are kind of ugly in my eyes and should be avoided whenever possible (yes I know, I've done blocks in blocks both with Forms and Self Optimizing Block - but those were exceptions, ok).

Enough talk, let's see some code. I decided that one of the best approaches would be if we could let the developers decide which properties should be inherited and how it should work. Something like this:

```
//Sidebar
[Inherit(InheritIfNullOrEmpty =true, ParentPropertyToInheritFrom ="DefaultChildSideBar", Se
public virtual ContentArea SideBar { get; set; }
```

What's happening here is basically just that on my Blog Post content type, I'm telling it that this property value should be inherited, if it's not set (null or empty). It should try to inherit from the parent page, if the parent page has a property called "DefaultChildSideBar". If I hadn't specified that, it would simply look for a parent property of the same name as the current property. I also tell it to search all ancestors - so if the parent doesn't have that property set, it should look to the grandparent and so on.

Sometimes you might like to let the editor decide if a value should be inherited or not - in that case, you could add another boolean property on the page and specify it's name as "SwitchPropertyName" in the attribute.

I haven't yet decided on a good strategy for when to populate the properties - so for now, I've let it be up to the developers - they basically have to call an extension method on IContent that will attempt to populate the needed inherited properties.

```
public static T PopulateInheritedProperties<T>(this T Content) where T : PageData
        {
            var rt = (Content as IReadOnly).CreateWritableClone() as PageData;
```

Blog posts

2019
November
October
September
August
July
June
May
April
March
February
January
2018
December
November
October
September
August
June

Available Topics

.NET DEVELOPMENT
ADDON DEVELOPMENT
AI
API BUILDING
AZURE
BEHAVIOR ANALYTICS
BIG DATA
C#
CMS
CONTENTFUL
DIGIZUITE
ELASTICSEARCH
EPISERVER
FREELANCING
INFORMATION RETRIEVAL
INTEGRATIONS
LIFE AS A CODER
MICRO SERVICES
OFFSHORE DEVELOPMENT
PRODUCT MANAGEMENT
TECH TALK
TIPS AND TRICKS
VISION DEMOS & PROTOTYPES

About Allan

I'm a freelance software architect / developer and Episerver expert (EMVP) based in Copenhagen Denmark. I've worked for Episerver for more than 10 years before going freelance and understand both your business needs as well as the technical possibilities.

epi MVP MOST VALUABLE PROFESSIONAL

I love solving real business problems with innovative and creative coding!

Learn more about my skills and specialities, reach out to me at info@codeart.dk or use this form to contact me.

```
            var props = Content.GetPropertiesWithAttribute(typeof(InheritAttribute));
            bool modified = false;
            foreach (var prop in props)
            {
                var attr = prop.GetCustomAttribute<InheritAttribute>(true);

                if (
                    (!String.IsNullOrEmpty(attr.SwitchPropertyName) && ((bool)Content.GetTy
                    ((attr.InheritIfNull || attr.InheritIfNullOrEmpty) && (prop.GetValue(Co
                    (attr.InheritIfNullOrEmpty && ((prop.PropertyType == typeof(ContentArea
                    )
                {
                    //Resolve Inherited Properties
                    var repo = ServiceLocator.Current.GetInstance<IContentRepository>();
                    foreach(var a in repo.GetAncestors(Content.ContentLink).Take((attr.Sear
                    {
                        var parentprop = (a as IContentData).Property[attr.ParentPropertyTo
                        if (parentprop!=null && !parentprop.IsNull)
                        {
                            prop.SetValue(rt, parentprop.Value);
                            modified = true;
                            break;
                        }
                    }
                }
            }
            if (modified)
            {
                rt.MakeReadOnly();
                return rt as T;
            }
            return Content;

        }
```

The attribute presents these options:

```
    public class InheritAttribute : Attribute
    {
        /// <summary>
        /// Name of Boolean property that indicates if this property should be inherited
        /// </summary>
        public string SwitchPropertyName { get; set; }

        /// <summary>
        /// Inherit this value if it's null
        /// </summary>
        public bool InheritIfNull { get; set; }

        /// <summary>
        /// Inherit this value if it's null or empty
        /// </summary>
        public bool InheritIfNullOrEmpty { get; set; }

        /// <summary>
        /// Name of property on parent content to inherit from. Default is same name.
        /// </summary>
        public string ParentPropertyToInheritFrom { get; set; }

        /// <summary>
        /// Keep searching ancestors until Root
        /// </summary>
        public bool SearchAllAncestors { get; set; }

    }
```

You can see the entire Gist below.

A word of caution

This is in no way a done solution - in fact, it's a very first draft. I just figured I'd share it here to get some feedback (which is very welcome in the comments below).

There are still many pieces to the puzzle missing. For example:

- What about Blocks? What will they inherit from?
- Should we also try to resolve inheritance recursively on parents with inherited properties?
- When should we resolve? Currently I resolve inherited properties by calling the method in my Controller.
- How can we alter the UI to make the editor aware that a certain property is begin inherited - and from where it's being inherited?

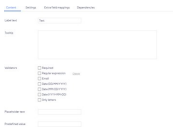
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5
6  namespace AllanTech.Web.PropertyInheritance
7  {
8      public class InheritAttribute : Attribute
9      {
10         /// <summary>
11         /// Name of Boolean property that indicates if this property should be inherited
12         /// </summary>
13         public string SwitchPropertyName { get; set; }
14
15         /// <summary>
16         /// Inherit this value if it's null
17         /// </summary>
18         public bool InheritIfNull { get; set; }
19
20         /// <summary>
21         /// Inherit this value if it's null or empty
22         /// </summary>
23         public bool InheritIfNullOrEmpty { get; set; }
24
25         /// <summary>
26         /// Name of property on parent content to inherit from. Default is same name.
27         /// </summary>
28         public string ParentPropertyToInheritFrom { get; set; }
29
30         /// <summary>
31         /// Keep searching ancestors until Root
32         /// </summary>
33         public bool SearchAllAncestors { get; set; }
34
35     }
36 }
```

InheritAttribute.cs hosted with ♥ by GitHub view raw

```cs
using EPiServer.Core;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AllanTech.Web.Helpers;
using EPiServer.ServiceLocation;
using EPiServer;
using EPiServer.Data.Entity;
using System.Reflection;

namespace AllanTech.Web.PropertyInheritance
{
    public static class PropertyInheritor
    {

        public static T PopulateInheritedProperties<T>(this T Content) where T : PageData
        {
            var rt = (Content as IReadOnly).CreateWritableClone() as PageData;
            var props = Content.GetPropertiesWithAttribute(typeof(InheritAttribute));
            bool modified = false;
            foreach (var prop in props)
            {
                var attr = prop.GetCustomAttribute<InheritAttribute>(true);

                if (
                    (!String.IsNullOrEmpty(attr.SwitchPropertyName) && ((bool)Content.GetType(
                    ((attr.InheritIfNull || attr.InheritIfNullOrEmpty) && (prop.GetValue(Conte
                    (attr.InheritIfNullOrEmpty && ((prop.PropertyType == typeof(ContentArea))
                    )
                {
                    //Resolve Inherited Properties
                    var repo = ServiceLocator.Current.GetInstance<IContentRepository>();
                    foreach(var a in repo.GetAncestors(Content.ContentLink).Take((attr.SearchA
                    {
                        var parentprop = (a as IContentData).Property[attr.ParentPropertyToInh
                        if (parentprop!=null && !parentprop.IsNull)
                        {
                            prop.SetValue(rt, parentprop.Value);
                            modified = true;
                            break;
                        }
                    }
                }
            }
            if (modified)
            {
                rt.MakeReadOnly();
                return rt as T;
            }
            return Content;

        }


    }
}
```

PropertyInheritor.cs hosted with ♥ by GitHub view raw

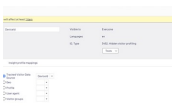
Blog Posts



November 21 2019 | TIPS AND TRICKS, C#, CMS, EPISERVER

Episerver Forms: Adding custom validators to existing elements

It's easy to extend Episerver forms with custom validation types. But it's a little bit harder to add the new validation types to the existing elements. In this blog post I'll add an ultra simple letter-only validation option to the existing TextboxElement.



November 6 2019 | TIPS AND TRICKS, EPISERVER

Episerver Forms: Adding Datasource for hidden field with Profile Store data

Some times, it can be handy to track additional data about the visitor along with a form submission. If you have Profile Store, stuff like DeviceId, sessionId and maybe even the ProfileId can definitely come in handy later when you are later processing the form submissions. Here is a code sample on how to do that.

October 29 2019 | TIPS AND TRICKS, TECH TALK, CMS, EPISERVER

Ascend 2019: Code Mania

Yesterday, I had the honor and pleasure of giving the traditional Code Mania demo at Episerver Ascend 2019 in Miami together with Fredrik Haglund. After popular demand, here is a blog post about some of the components we showed.

August 29 2019 | TIPS AND TRICKS, CMS, BEHAVIOR ANALYTICS, EPISERVER

Getting more Insight (pun intended) into Episerver Profile Store

Profile Store, Insight, Tracker, Advance - Episerver offers a myriad of different (but connected) REST services for managing and tracking your visitors and prospects. It can be slightly confusing at first - and some of the

August 13 2019 | TIPS AND TRICKS, INTEGRATIONS, CMS, EPISERVER

Episerver Forms and Pardot Form Handlers

Episerver comes with a wide range of connectors that allows you to connect Marketing Automation systems to Episerver through multiple integration points. However in some cases you might want to hook directly into the Marketing Automation systems form handler. In the case of SalesForce Pardot it's very

July 8 2019 | TIPS AND TRICKS, EPISERVER

The tricky redirect

Sometimes, when you are troubleshooting you forget the obvious in search of more complex reasons. Recently I had a case of that where an Episerver site kept redirecting the moment it was launched.

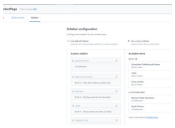
documentation might be a tad misleading - but once you get the hang of it, they are really powerful tools. I've recently had a chance to explore them in depth. Here is what I've learned so far.

easy to do!

July 3 2019 | TIPS AND TRICKS, EPISERVER

Site Map Generation with Custom Filter

There's a handful of great add-ons I use in almost every Episerver project - and several of them are from Geta. Here is a useful hint if you, like me, use the Geta SiteMap generator.



July 2 2019 | ADDON DEVELOPMENT, CONTENTFUL, TECH TALK, CMS

Getting started with Contentful UI Extensions - Part 3

Sidebar extensions is a great way to add tools, widgets and integrations to editors, without relying on a specific field. In this post I'll explore them a little, and also test out how much crazy stuff we can actually do with the javascript SDK.

June 19 2019 | CONTENTFUL, ADDON DEVELOPMENT, TECH TALK, CMS

Getting started with Contentful UI Extensions - part 2

In this post, I'll show how to make a field editor that will let you have any kind of syntax highlighted code in a long text field, as well as taking a look at command line interface (CLI) and Github distribution.

June 17 2019 | INTEGRATIONS, EPISERVER, DIGIZUITE, ADDON DEVELOPMENT

Digizuite DAM for Episerver - try it out!

The Digizuite DAM for Episerver integration is now available in the Episerver nuget feed! We've worked long and hard on this, so feel free to have a look and try it out!

June 12 2019 | CONTENTFUL, CMS, TECH TALK, INTEGRATIONS, ADDON DEVELOPMENT

Getting started with Contentful UI Extensions - Part 1

Contentful has a handful of extension points, where you in a fairly straightforward and simple manner can extend the editorial experience with minimal development effort. In this post-series I'll show some examples of this.

May 9 2019 | INFORMATION RETRIEVAL, EPISERVER, CMS

Publicwww - searching for interesting Episerver CMS use patterns

I recently discovered publicwww.com a cool service that lets you search for any text in the html/css/js of all it's 550 million (2019-05-09) indexed web pages, including the cookies sent out and the http header. In this post I put my Episerver goggles on and had some fun with this data.

‹ 1 2 3 4 5 ›

C# EPISERVER

Previous Post
Content Report Generator

Next Post
Starting out on my own

Post Comments (3)

CodeArt ApS
Frederiksdalsvej 198, 2830 Virum, Denmark
Phone: +45 26 13 66 96
CVR: 39680688

Tweets by athraen