Fallback property values in EPiServer using attributes

Fallback property values in EPiServer is a common, and often repetitive, requirement which can be simplified by decorating content type properties with an attribute to specify fallback behavior.

Common approach for property fallback values

Fallback values for content type properties are often implemented by **overriding property getters and setters**. While this is an effective way of implementing specialized logic for retrieving the fallback value, it’s a bit cumbersome, verbose and repetitive for many of the more trivial cases:

```
public virtual string Title
{
    get
    {
        var title = this.GetPropertyValue(page => page.Title);

        if (!string.IsNullOrWhiteSpace(title))
        {
            return title;
        }

        // Fallback to page name when title isn't set
        return PageName;
    }
    set
    {
        this.SetPropertyValue(page => page.Title, value);
    }
}
```

Revised approach based on attribute decoration

With our revised approach we can achieve the same result as before by decorating our property with a **Fallback attribute**:

```
[Fallback(PropertyName = "PageName")]
public virtual string Title { get; set; }
```

Additional fallback options

We can specify a fallback value **explicitly**, for example to specify a default author:

```
[Fallback(Value = "Kevin Flynn")]
public virtual string Author { get; set; }
```

We could use this in combination with a **PropertyName** parameter, using the default author name only if the fallback value is also empty:

```
[Fallback(PropertyName = "CompanyName", Value = "Kevin Flynn")]
public virtual string Author { get; set; }
```

If our fallback value comes from a nested property of a complex type, like a **local block**, we can use **dot notation** for the property name:

```
[Fallback(PropertyName = "MyBlockProperty.CompanyName", Value = "Kevin Flynn")]
public virtual string Author { get; set; }
```

The examples so far assume the fallback property is part of the **same content instance** as the original property.

If our fallback property is defined on **another content instance**, we can specify that the fallback content will be determined by a **ContentReference** property.

The following would use a **property** of a **local block** on the **start page** as the fallback value:

```
public class MyPageType: PageData
{
    [Fallback(PropertyName = "SiteSettings.CompanyName", ContentReferencePropertyName = "SettingsPage", Value = "Kevin Flynn")]
    public virtual string Author { get; set; }

    public ContentReference SettingsPage
    {
        get { return ContentReference.StartPage; }
    }
}
```

How it works

EPiServer intercepts all **getters** and **setters** of content type properties using an instance implementing **IInterceptor**, part of the **Castle Project**. The default interceptor in EPiServer is called **ContentDataInterceptor**, but using EPiServer's **IoC** container we can easily replace this with a custom interceptor which supports our **Fallback** attribute.

Creating the Fallback attribute

The **Fallback** attribute is quite trivial:

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = true)]
public class FallbackAttribute : Attribute
{
    public virtual string ContentReferencePropertyName { get; set; }

    public virtual string PropertyName { get; set; }

    public virtual object Value { get; set; }
}
```

Switch concrete implementation to a custom interceptor

One way to switch concrete implementations in EPiServer is to create a class implementing **IConfigurableModule**, an interface very similar to **IInitializableModule** with the addition of a **ConfigureContainer** method:

```
[InitializableModule]
[ModuleDependency(typeof(ServiceContainerInitialization))]
public class FallbackInitialization : IConfigurableModule
{
    public void Initialize(EPiServer.Framework.Initialization.InitializationEngine context) { }

    public void Preload(string[] parameters) { }

    public void Uninitialize(EPiServer.Framework.Initialization.InitializationEngine context) { }

    public void ConfigureContainer(ServiceConfigurationContext context)
    {
        // Register custom interceptor
        context.Container.Configure(config => config.For<ContentDataInterceptor>()
                                            .Use<FallbackValueContentDataInterceptor>());
    }
}
```

Implementing the custom interceptor

This is where the magic happens. We inherit **ContentDataInterceptor** and add logic which uses the **Fallback** attribute to determine how a fallback value should be retrieved. This code is fairly lengthy, but the comments hopefully clarify what is going on:

```
using System;
using System.Reflection;
using Castle.DynamicProxy;
using EPiServer;
using EPiServer.Core;
using EPiServer.DataAbstraction.RuntimeModel;
using EPiServer.ServiceLocation;
using log4net;

namespace TedGustaf.Web.PropertyFallback
{
    /// <summary>
    /// Enables fallback attributes for content type properties
    /// </summary>
    /// <remarks>Configured through <see cref="FallbackInitialization"/></remarks>
    /// <author>Ted Nyberg, @tednyberg</author>
    public class FallbackValueContentDataInterceptor : ContentDataInterceptor
    {
        private static readonly ILog _logger = LogManager.GetLogger(typeof (FallbackValueContentDataInterceptor));

        protected override void HandleGetterAccessor(IInvocation invocation, PropertyData propertyData)
        {
            base.HandleGetterAccessor(invocation, propertyData);
            if (!IsNull(invocation.ReturnValue)) // Property value is set
            {
                return;
            }

            // Get the property definition of the content type, i.e. a property member with the same name as the content type prope
            var propertyProperty = invocation.InvocationTarget.GetType().GetProperty(propertyData.Name);

            if (propertyProperty == null)
            {
                _logger.WarnFormat("There is no property called {0} on type {1}, content type property does not map to model type",

                return;
            }

            // Get the fallback attribute, if any, decorating the property
            var fallbackAttribute = Attribute.GetCustomAttribute(propertyProperty, typeof (FallbackAttribute), true) as FallbackAtt

            if (fallbackAttribute == null) // No fallback attribute
            {
                return;
            }

            // Get the fallback value based on the fallback attribute parameters
            invocation.ReturnValue = GetValue(invocation.InvocationTarget, propertyProperty, fallbackAttribute);
        }

        /// <summary>
        /// Throw an exception if the attribute parameters are invalid
        /// </summary>
        protected virtual void ThrowOnInvalidAttributeParameters(FallbackAttribute attribute, PropertyInfo property)
        {
            if (string.IsNullOrWhiteSpace(attribute.ContentReferencePropertyName)) // Fallback value will be retrieved from current
            {
                if (string.IsNullOrWhiteSpace(attribute.PropertyName) && attribute.Value == null) // Neither property name nor fixe
                {
                    throw new NotSupportedException("Fallback value attribute must specify either a content reference property name
                }

                if (!string.IsNullOrWhiteSpace(attribute.PropertyName) && attribute.PropertyName.Equals(property.Name)) // Fallback
                {
                    throw new NotSupportedException("Fallback property cannot be the same as the source property when no content re
                }
            }

            if (attribute.Value != null && attribute.Value.GetType() != property.PropertyType) // The specified fallback value does
            {
                throw new InvalidCastException(string.Format("The explicit fallback value is of type {0}, but the property type is
            }
        }

        /// <summary>
        /// Gets the property value, or fallback value based on fallback attribute parameters
        /// </summary>
        protected virtual object GetValue(object instance, PropertyInfo property, FallbackAttribute attribute)
        {
            _logger.DebugFormat("Retrieving fallback value for '{0}' for instance of type {1}", property.Name, instance.GetType().N

            object value = null;

            ThrowOnInvalidAttributeParameters(attribute, property);

            ContentReference fallbackValueContentReference = null;

            if (!string.IsNullOrWhiteSpace(attribute.ContentReferencePropertyName)) // A content reference property on the instance
            {
                // Resolve the content reference property on the current content instance
                var fallbackValueContentReferenceProperty = ResolveProperty(attribute.ContentReferencePropertyName, instance);

                if (fallbackValueContentReferenceProperty == null) // Content reference property not found
                {
                    throw new NotSupportedException(string.Format("The content type {0} does not have a property called {1}", insta
                }

                fallbackValueContentReference = fallbackValueContentReferenceProperty.GetValue(instance) as ContentReference;

                if (fallbackValueContentReference == null) // Content reference property is an incorrect type
                {
                    throw new NotSupportedException(string.Format("The property named '{0}' is not a ContentReference", attribute.C
                }
            }

            if (!ContentReference.IsNullOrEmpty(fallbackValueContentReference)) // Get fallback value from content instance specifi
            {
                var fallbackValueContentData = ServiceLocator.Current.GetInstance<IContentLoader>().Get<ContentData>(fallbackValueC

                if (string.IsNullOrWhiteSpace(attribute.PropertyName)) // Fallback property name not specified
                {
                    attribute.PropertyName = property.Name; // Use property of same name from fallback content instance
                }

                var fallbackProperty = ResolveProperty(attribute.PropertyName, fallbackValueContentData);

                if (fallbackProperty == null) // Specified property name does not exist on fallback content instance
                {
                    _logger.WarnFormat("Fallback content instance '{0}' does not have a property called '{1}'", fallbackValueConten
                }
                else
                {
                    if (attribute.PropertyName.Contains(".")) // Nested fallback property, i.e. property of a complex property type
                    {
                        var nestedPropertyInstance = ResolveInstance(attribute.PropertyName, fallbackValueContentData);

                        value = fallbackProperty.GetValue(nestedPropertyInstance);
                    }
                    else
                    {
                        value = fallbackProperty.GetValue(fallbackValueContentData);
                    }
                }
            }
            else if (!string.IsNullOrWhiteSpace(attribute.PropertyName)) // An fallback property name has been specified
            {
                var fallbackProperty = instance.GetType().GetProperty(attribute.PropertyName);

                if (fallbackProperty == null)
                {
                    _logger.WarnFormat("Current instance does not have a property called '{0}'", attribute.PropertyName);
                }
                else
                {
                    value = fallbackProperty.GetValue(instance);
                }
            }

            return value ?? attribute.Value; // Use explicit fallback value if no other fallback value could be found
        }

        /// <summary>
        /// Resolves the instance containing the specified nested property
        /// </summary>
        /// <param name="sourceInstance">The original content instance for which the nested instance should be resolved</param>
        /// <param name="propertyIdentifier">Dot notation to specify a nested property, like MyType.MyComplexProperty.NestedPropert
        /// <returns></returns>
        private object ResolveInstance(string propertyIdentifier, object sourceInstance)
        {
            if (sourceInstance == null)
            {
                throw new ArgumentNullException("sourceInstance", "No source instance specified, unable to resolve nested property
            }

            if (string.IsNullOrWhiteSpace(propertyIdentifier))
            {
                throw new ArgumentNullException("propertyIdentifier", "No property identifier specified, unable to resolve instance
            }

            if (!propertyIdentifier.Contains("."))
            {
                throw new ArgumentException("Property identifier must be in dot notation to resolve nested property");
            }

            var segments = propertyIdentifier.Split('.');

            object instance = sourceInstance;

            // Resolve instances up until the final segment, i.e. the property for which the instance should be resolved
            for (int i = 0; i < segments.Length - 1; i++)
            {
                var segment = segments[i];

                var property = ResolveProperty(segment, instance);

                if (property == null)
                {
```

```
                _logger.WarnFormat("Unable to find property {0} on type {1}", segment, instance.GetType().Name);

                return null;
            }

            instance = property.GetValue(instance);
        }

        return instance;
    }

    /// <summary>
    /// Resolves a property on the specified instance, including dot notation to support nested properties
    /// </summary>
    /// <returns>Null if the property cannot be resolved</returns>
    protected virtual PropertyInfo ResolveProperty(string propertyIdentifier, object instance)
    {
        if (string.IsNullOrWhiteSpace(propertyIdentifier))
        {
            throw new ArgumentNullException("propertyIdentifier", "No property identifier specified");
        }

        PropertyInfo property = null;

        if (!propertyIdentifier.Contains("."))
        {
            property = instance.GetType().GetProperty(propertyIdentifier);
        }
        else
        {
            var identifierSegments = propertyIdentifier.Split('.');

            foreach (var segment in identifierSegments)
            {
                // Get property from original instance, or from nested property within it
                if (property != null)
                {
                    instance = property.GetValue(instance);
                }

                property = ResolveProperty(segment, instance);

                if (property == null)
                {
                    _logger.WarnFormat("No property called {0} on type {1}", segment, instance.GetType().Name);

                     return null;
                }
            }
        }

        if (property == null)
        {
            _logger.WarnFormat("Unable to resolve property using identifier '{0}' on content type {1}", propertyIdentifier, ins
        }
```

```
            return property;
        }

        /// <summary>
        /// Checks if a value is null, or should otherwise trigger fallback behavior
        /// </summary>
        protected virtual bool IsNull(object value)
        {
            // TODO Check for boundary DateTime etc that should trigger fallback behavior?

            return value == null || (value is string && string.IsNullOrWhiteSpace(value as string));
        }
    }
}
```

Notes of interest

Most aspect-oriented libraries, like **PostSharp**, can be used to easily extend support for the **Fallback** attribute to intercept all properties, not just content type properties.

Disclaimer

The code is a first prototype draft and is provided as-is under MIT license.

← Previous Next →

0 Comments Ted & Gustaf 1 Login

Recommend Share Sort by Best

Start the discussion…

Be the first to comment.

Subscribe Add Disqus to your site Privacy

Ted Nyberg
CEO and Founding Partner
+46 707 770 790
ted@tedgustaf.com

Twitter

LinkedIn

Tags

- episerver
- code samples

Share it if you like it

Tweet 5
Gilla 0

Articles on EPiServer

More from the blog

Keep in touch

- EPiServer Google Maps editor on NuGet
- Fallback property values in EPiServer using attributes
- Google Maps custom editor for EPiServer 7.5
- Add a custom toolbar button in EPiServer 7
- Create an EPiServer widget for edit mode
- Usability testing of click behavior
- Web Essentials plugin for Visual Studio 2013
- Web Essentials plugin for Visual Studio 2013
- Compile .less files online
- Create Excel 2007/2010 spreadsheets with C# and EPPlus

Phone	+46 (8) 410 208 40
E-mail	info@tedgustaf.com
Address	Sveavägen 21 111 34 Stockholm