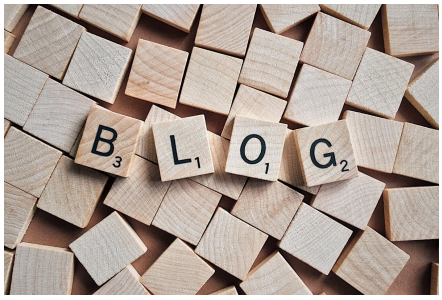
ALLAN THRAEN | 15 years ago | PDF |

# AUTO LOADING OF DYNAMIC CONTENT




One of the really cool new features in EPiServer CMS 5 R2 is in my humble opinion Dynamic Content. It brings with it a lot (!) of potential. Together with PageProviders we might have something here that could turn out to be the solution to most major problems we face (ok - besides world hunger, cancer, etc) - it's really just a matter of using it creatively and wisely. R2 is rapidly reaching it's 1 week anniversary and it seems like it's being really well received in the market so far. However I have heard one or two whine about having to register DynamicContent blocks in the web.config (as opposed to the traditional EPiServer plugin attribute) before using them. Registering providers and plugins in web.config is a rapidly emerging trend that I suspect we'll see a lot more of in the future. **It's good for performance** and gives the developer / administrator good control over all the websites settings in one place. That being said I can still understand the concerns of developers wanting the easy deployment of modules that the attribute gave them - and I suppose it could be an interesting challenge to demonstrate how one can makes his own attribute that auto-registers Dynamic Content.

The approach is simple. As I've blogged about before any class inheriting from PluginAttribute will have a static method named "Start" executed when the web application initializes. So, why not create a new attribute inheriting from PluginAttribute and have it's static start method find all DynamicContent with that attribute and add it to the DynamicContent configuration? Well, unfortunately it's not as simple as that - because of the order of an EPiServer's initialization.

Here is the main initialization method (extracted by Reflector):

```
1:  public static void StaticInitialization()
2:  {
3:      lock (_setupLock)
4:      {
5:          Settings.Instance.ValidateRuntimeSettings();
6:          InitializeEnterpriseLicense();
7:          Initialize(EPiServerSection.Instance, Settings.Instance);
8:          DataFactoryCache.AllowUploadEndpoints();
9:          LazyIndexer.Init(DataFactory.Instance, Settings.Instance.IndexingDelayAfter
10:         PlugInRuntime.Start();
11:     }
```

The Configuration is loaded in the Initialize call, but the plugin-attributes are not called until the PlugInRuntime.Start(), meaning that whatever we add to the configuration at that stage will not be used to initialize the DynamicContent.

The solution is of course to add our DynamicContent elements directly to the DynamicContentFactory. It contains two mapping tables that maps name to type and back again - and adding the elements there is fairly simple. However sometimes more meta-information about a DynamicContent element is needed on top of it's name and it's type (like the description) - so we also need to add it to the configuration anyway - but then we're good to go.

My code ended up looking like this:

```
1:  using System;
2:  using System.Configuration;
3:  using EPiServer.PlugIn;
4:  using EPiServer.Configuration;
5:
6:  namespace EPiServer.Research.DynamicContent
7:  {
8:      public class DCPlugin : PlugInAttribute
9:      {
10:         public static void Start()
11:         {
```

Left is just to add the attribute to the registration of your DynamicContent classes and set the DisplayName to be the name of your DynamicContent like this:

[

IDynamicContent

DCPlugin(DisplayName="HelloWorld",Description="Displays a Hello World text")]publicclassHelloWorld:_

This basic approach can probably be used to register other providers & plugins as well - but use with caution and on your own risk :-)

Released under EPiOpen license.

RECENT POSTS

**CodeArt ApS**
Teknikerbyen 5, 2830 Virum, Denmark
**Email:** info@codeart.dk
**Phone:** +45 26 13 66 96
CVR: 39680688