ALLAN THRAEN | 15 years ago | PDF |

BUILDING YOUR OWN PAGE PROVIDER: NORTHWIND

EPiServer CMS 5 R2 CTP is awesome! I've now finally found a few hours to play around with some of the new cool functionality and I really like what I see. In particular I've grown fond of custom page providers. It's extremely cool to be able to connect more or less anything to an EPiServer and have it appear as pages in the structure. To try it out I've made a couple of providers based on the public CTP (can be installed from the Manager if you have a developer license).

Here's one of them: A "Northwind" product page provider. Some of you might remember the infamous "Northwind" database - the sample database that for many years was included in any SQL Server installation and without which no microsoft textbook programming example was complete. It isn't that popular anymore, however it's easy to find somewhere on the big fuzzy thing we all know as *the internet* :-)

My goal with this first provider is to simulate an online shop where there is an already existing product database that should be exposed in EPiServer without duplicating the data. I'll use the products and product categories from Northwind as example data. To make everything a bit easier I've used .NET 3.5 with LINQ to work with the database.

Getting started

In order to work with custom page providers an enterprise license is needed. When you request a partner developer license you can fill in a comment that it needs to be an enterprise developer license. When the R2 installation is up and running and the correct license is installed you should decide where in your page structure you want to attach the pages from the custom page provider (you can also replace the entire tree by making the new provider the default provider, but that's not my approach in this post). Instead I created a new page on my site and decided that it should be the root of my custom provider, so I made a mental note of it's page-id.

A custom page provider basically just requires two things to work: the class that defines the provider itself (which should inherit from PageProviderBase and implement a minimum of 4 it's methods that's marked as abstract) and a registration in web.config where you define the type, name and entrypoint (parent page-id) of the provider.

GetChildrenReferences and GetLocalPage

The two most interesting methods you need to make in your provider are the GetChildrenReferences and GetLocalPage.

In **GetChildrenReferences** you get a PageReference to a parent page and you should return a PageReference collection with references to all it's children. Typically the first call to this method will be with the reference to the EntryPoint, the page in your default provider you decided to use as a kind of a "mounting point" for your custom provider.

You can check if this is the case by seeing if it's equal to the value you store in "base.EntryPoint" of your provider.

In my Northwind provider I want two levels of pages. As children to the EntryPoint I want the product categories from Northwind, and below each category I want the products in that category. So in this method I basically make an if-statement that handles the scenarios.

```
protected override PageReferenceCollection GetChildrenReferences(PageReference pageLink, strin
{
    PageReferenceCollection prc = new PageReferenceCollection();
    if (pageLink == base.EntryPoint)
    {
        //List product categories
        foreach (EPiServer.PageProviders.Category c in ListCats())
        {
            prc.Add(base.ConstructPageReference(c.CategoryID));
        }
    }
    else if(pageLink.ID<100)
    {
        //We have a category selection
        foreach (Product p in ListProducts(pageLink.ID))
        {
            prc.Add(base.ConstructPageReference(p.ProductID + 100));
        }
    }
    return prc;
}
```

You will notice that I use the size of the ID to determine if it's a category or a product - that is one way of combining several different id values into one (in my case I both have the ID's from the Categories table and from the Products table in Northwind). I use a helper method on the base object to construct a page reference that also includes the setting indicating that this ID belongs to my custom provider. The ListCats and ListProducts methods are from a plain and boring LINQ based datalayer that I'm not going to bore you with here - download the code if you want to see it.

GetLocalPage is where you on-the-fly create a PageData object containing the data you want to show for a given pagereference (that points to a page in your providers domain). Again, there is a nice little helper method in the base class that will help you initialize your PageData object after you create it with all the mandatory settings. One thing you should pay particular attention to here is to make sure that you set a PageReference to the Parent of the current page, that corresponds to the structure you are also exposing in the GetChildrenReferences method! In my case that means that when I'm building a PageData for a Category I set the parent to point to the base.EntryPoint, and when I'm building a PageData for a product I point to the products category. Another good idea here can be to set the PageURLSegment property to a name that's suitable for friendly urls - otherwise you might get some really ugly urls.

```
protected override PageData GetLocalPage(PageReference pageLink, ILanguageSelector languageSel
{
    PageData pd = new PageData();
    //If product, parent is category - otherwise entrypoint
    if (pageLink.ID < 100)
    {
        EPiServer.PageProviders.Category c = LookupCategory(pageLink.ID);
        base.InitializePageData(pd, pageLink, base.EntryPoint);
        pd.SetValue("PageName", c.CategoryName);
        pd.SetValue("MainBody", c.Description);
    }
    else
    {
        Product p = LookupProduct(pageLink.ID - 100);
        base.InitializePageData(pd,  pageLink, base.ConstructPageReference(p.CategoryID.Value)
        pd.SetValue("PageName", p.ProductName);
        pd.SetValue("MainBody", p.QuantityPerUnit+": "+p.UnitPrice.ToString());
    }
    SetPageStatus(pd, VersionStatus.Published);
    string seg = Regex.Replace(pd.PageName.Replace(' ', '-'), @"[^A-Za-z0-9\-_~]{1}", "");
    pd.SetValue("PageURLSegment", seg);
    return pd;
}
```

The *base.InitializePageData(...)* method requires quite a lot of parameters that can require some logic to determine, so for now I've made my own simple overload in an intermediate base class that I've used for this example. You'll learn more about why now...

Tricky stuff: Guids, ID's and resolving local pages

EPiServer CMS needs both a Guid and an Integer ID for every page. The ID should be unique within the provider and is used for making PageReferences, the Guid should be unique across all pages and is among other things used to maintain links so they won't break even if you move a page.
If you create your own new data storage from scratch and can design the data structure as you please it's usually no problem to store both numbers for every item you want to have appear as a page. However in many cases I imagine that you'd quite often use this functionality to expose data from already existing data stores: databases, order systems, file system, active directory, etc. where you can't easily get a Guid if there isn't one already. At least that was the problem I was facing with my Northwind Provider. In the code-pieces above you'll see that I to some degree solved the ID/PageReferences problem by using the ID fields already existing in my db by simply combining them by pushing some of them up with the value of 100 to make room for the others. However I still don't have a Guid. And for your provider you'll need a Guid for each page that you can lookup later. Being lazy and halfway on vacation I took the easy way out and created a few helper methods to make up a Guid from an ID - and also being able to easily later determine if a Guid came from this provider.

The reason for the need to determine if a given Guid belongs to your provider is quite simple: You also needs to implement method(s) that allows EPiServer to resolve links from either PageReferences or Guids. These methods will be called even for pages that are not in your provider and you need to be able to lookup if the guid / PageReference "belongs" to you or not. If not you should just return null and it will (hopefully) be resolved by another Page Provider.

Since I could see potential here for reusing some code in several providers I made a "GuidlessProvider" base class that implements these methods along with methods for working with Guids, etc.

Final result

After a fun hour of coding and a couple of hours troubleshooting here you have your first custom page provider.

All that is left for you to do is to implement it in your web.config:

```
    <pageProvider>
      <providers>
    <add name="NW"
type="EPiServer.PageProviders.NorthwindProductProvider,EPiServer.Templates.Public" entryPoint="30"/>
      </providers>
    </pageProvider>
```

It works like a charm, however you'll still need to implement a handful of methods if you want to be able to Edit, Create, Move, Delete, Search, etc. But I'll leave that for a future posting :-)

RECENT POSTS

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