ALLAN THRAEN | 16 years ago | PDF |

MAPPED PAGE PROVIDER

Until now, my favorite new feature in EPiServer CMS 5 R2 is the ability to attach custom page providers.

It's truly awesome to be able to connect any kind of external data source to your CMS and both see and manage the data as if it's ordinary EPiServer pages. I've already begun to make a bunch of different page providers for pretty much anything (AD, Filesystem, Commerce Server, SQL Server, EPiServer Profiles, Community, …) – and there is definitely endless potential for its uses. However, until now I have had a small stone in my shoe whenever I was making a page provider. When inheriting from PageProviderBase, you have to manually map and handle guids, ids and urls. That's a pain. Not only is it a lot of work, where it's easy to introduce errors – but it can also be quite tricky to store the guids and ids for an external data source that doesn't have those fields. Sure, if you're making a XML provider like the sample one listed, it's no problem to add a couple of extra attributes with a generated guid and id. But when you're working with external data like the company product database, the user profiles stored in active directory, etc. it's not particularly elegant to modify the external data in order to use it in your EPiServer.

So – I've started to make a new base class that handles that problem: The MappedPageProvider. The idea is that instead of working with id and guid you just have to map to one external string key. In order to do this it adds a table to the EPiServer database called tblPageProviderMappings, that it uses to keep track of all mappings. Once you start using the MappedPageProvider, it will add the table if it doesn't exist.

Here's an example of a simple hardcoded page provider implemented using the MappedPageProvider as a base:

```
public class TestProvider : MappedPageProvider
{
    protected override List<string> GetRootChildren()
    {
        return new List<string>() { "Allan", "Knudsen", "Kolle" };
    }

    protected override List<string> GetChildren(string Key)
    {
        if (Key == "Allan")
        {
            return new List<string>() { "Maximilian" };
```

First, you need to list the string keys to the root children (the child pages of the providers entry point). In this example I've chosen to use real names instead of a more useful id – it just gives that cozy feel :-) I've also implemented the GetChildren method, where you get a key to a page and return a list of child-keys from that page. In this case there is only one child. We also need a method that can return a PageData for a given string key. A helper method for that is InitializePageData – which now has an overload that supports using keys. In this case I also choose to use the key as the page name. Finally, a method that can identify which PageType to use for a given key is also needed. All there is left is just to compile and register the provider in web.config.

The "Test" node is in the local provider, and is set as the entryPoint.

Limitations

This Mapped Page Provider is just a first draft of a prototype. There is no support yet for saving, moving, deleting, searching and handling multiple versions / languages.

UPDATE 27 APR 2009: Download updated with some bug-fixes, support for saving, and based on EPiServer CMS 5 R2 SP1.

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