ALLAN THRAEN | 15 years ago | PDF |

NON-CACHED, CROSS PAGETYPE, GENERIC PAGE PROPERTIES

There are many different ways of storing business data in EPiServer CMS. And lots of opinions on them as well. David Hunt has made an excellent blog post about some of the approaches. On top of those approaches there are also others – like using the "secret" ObjectStore – or this approach I built a prototype of some time ago (but just recently found again during a bit of cleanup).

Today you basically have 2 kinds of properties in EPiServer CMS. You have Page Type properties and Dynamic properties. Page Type properties are defined on the page types, dynamic are inherited across all pages. However, I felt that in many cases it would also come in handy to on-the-fly add undeclared simple properties to individual **pages**. Properties that didn't apply to any other page than the one it was actually added on.

My need at the time was a little tool I was working on that would allow you to store editor comments on specific parts of a page (a bit like post-its) for other editors to see (will blog about this sometime in the future), but after I made it, I realized that the code could also be used for other frequently changing data. Like a counter of page-views. Or a simple article rating system. The thing is that the pagetype properties aren't optimized for rapidly changing values (they require a full page publish, which has a lot of cache implications). Instead, these lightweight properties, stored in their own db table, seemed to do the trick.

Basically the plug-in – which is a single assembly plug-in – will create a table of it's own, upon first initialization. The table basically stores a page reference, property name and property value. That's it. You access the properties, using extension methods to the PageData object as shown in this example that implements a basic page view counter:

```
using EPiServer.Research.PageProperties;

...

public string Counter()
{
    //Get the value for a page counter in a null-able int
    int? i = CurrentPage.PageProperty<int?>("Counter");
    if (!i.HasValue) i = 0;
    i++; //Increase the count
    CurrentPage.PageProperty<int?>("Counter",i); //Store the value
    return i.ToString();
```

There's still a lot of work to do on this – and it's only a very basic prototype I'm releasing here and now.

It only supports int and string for property values, the db table isn't optimized, it doesn't clean up properly, it has no UI, no version/multi language support, etc. but you might still see where I am going with this. I'd appreciate thoughts and input on this approach.

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