ALLAN THRAEN | 16 years ago | PDF |

PLAYING AROUND WITH LINQ AND EPISERVER CMS 5

I finally found a moment to begin playing around with EPiServer CMS 5 in conjunction with LINQ and the cool features of C# 3.0.

Ever since I first saw Anders Hejlsberg present his thoughts on LINQ several years ago, I've been dying to get my hands on it and play around. And sure, when the first tech preview was released I downloaded faster than you can say "lambda expressions". But the first previews (and the later ones) didn't always co-exist nicely with my other environment - so every time I played around with it a bit I always ended up uninstalling it.

But now it's been officially released with VS 2008, and I'm thrilled with it!

Naturally, I figured that it could be fun to see all the potential together with EPiServer CMS 5 - as well trying to figure out how to make EPiServer be *REALLY* integrated with LINQ.

There's a lot of ground to cover - and a lot of ideas waiting to be discovered here - so I'll just start out slow in this post, with a few simple examples. Hopefully I'll soon come up with more clever ones, so in the future I hope to really explore the possibilities within extension methods, lambda expressions, expression trees, type inference, anonymous classes and of course LINQ Queries against both XML, Databases and object structures. But for now here's just some random picks.

This is what I did to get going:

- I copied an EPiServer CMS 5 dummy website (KnowledgeWeb - those who've been on Developer course probably knows it).
- I opened the project in VS 2008 and let it convert the project
- I added a reference to System.Core.dll
- I added a new regular ASP.NET Page (since the EPiServer templates wasn't installed on my VS 2008)
- I changed the page to inherit from EPiServer.TemplatePage - to get the full functionality
- I added a gridview named "GridView1" to the page, and a couple of buttons.

My first simple test was to retrieve a filter on the child-pages to the Start Page, in order to only show the ones thats published - and in a simplified manner. This would allow me to try out type inference, LINQ query, anonymous classes and object initializers.

```
var plist = from p in GetChildren(PageReference.StartPage)
               where p.Status == VersionStatus.Published
               orderby p.PageName ascending
               select new { Name = p.PageName, Author = p.CreatedBy };
GridView1.DataSource = plist;
GridView1.DataBind();
```

I use type inference, using the "var" operator to infer the type of the "plist", and by writing "new {Name ... }" I create a new object of an anonymous class, with the properties Name and Author. Just for fun, I order by PageName descending - it's good with a bit of variation from the sort-order :-) Simple and nice.

The typical example on using LINQ queries on an object hierarchy is to do something cool with Reflection. And since I'm not better than all the other bloggers, my second code snippet goes like this:

```
Assembly a=Assembly.GetAssembly(typeof(PageData));
var typelist = from t in a.GetTypes()
               where t.IsSerializable
               orderby t.Name
               select new {
                   Name = t.Name,
                   FullName = t.FullName,
                   BaseType = t.BaseType.FullName }
            ;
GridView1.DataSource = typelist;
GridView1.DataBind();

Button2.Text = a.GetTypes().Count<Type>(t => t.IsSerializable).ToString();
```

Yes, thats right. It examines the "EPiServer.dll" and lists all the types that are Serializable, ordered by name - and just to make it ugly, calculates the count (using a Lambda expression) and put's it as the buttons text.

I will round off today's post with a small extension method example. Have you ever wondered why there isn't a PageBase.GetChildren() overload, that simply returns the current page's children? Well, the good news is that now you can make that method yourself using Extension Methods.

Here's what I did to get it to work:

```
    public static class PageExt
    {
        public static PageDataCollection GetChildren(this PageBase P)
        {
            return P.GetChildren(P.CurrentPage.PageLink);
        }
    }
```

You'll notice that the trick is to make a public, static method and add the type along with the "this" keyword as the first parameter. Try it for yourself!

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