ALLAN THRAEN | 16 years ago | PDF |

EPISERVER AND CUSTOM URLS USING ASP.NET URLROUTING

Ever wanted to introduce some special URL handling in your EPiServer web sites, like browsing pages per category by calling mysite.com/Category/[Category To Search for]? Well, rejoyce, cause now it's easier than ever before.

Since EPiServer CMS 5 R2 is based on .NET framework 3.5 SP1 we can now use the new UrlRouting module, in System.Web.Routing. Originally built as a part of ASP.NET MVC it has now been moved into the core part of the framework, and it can be used for a lot more than MVC. In fact, it works splendidly with Web Forms as well - or any other task you can think of. And the best part is that it happily co-exists with EPiServer's friendly URL handling.

Replacing the Url Rewriter?

"If it's so cool, why doesn't EPiServer simply replace the Url Rewriting with UrlRouting?" I hear you cry - well. My personal guess is that we might do that eventually - but it's not as straightforward as it seems. First of all the UrlRouting only solves half the problem since we still need to rewrite all outputted html to friendly urls - and secondly, a number of implementations already have either hooked into the url rewriting or made their own based on the original. But enough of me guessing what the product teams considerations are. Fact remains, that you can already get the benefits of UrlRouting today by following these simple steps.

Going Wiki

The example I'll show is from a project I'm currently working on - finalizing a Wiki for EPiServer (which will be totally awesome, btw). A wiki consist of a number of articles placed in a very friendly path structure. They could be placed like this */Wiki/Article1*, */Wiki/Article2*, ... No problems there, the Friendly Url rewriter handles that part splendidly. However, if a url is called to an article that isn't written yet - like */Wiki/NameOfNewArticle* we don't want to return a 404, but rather allow the visitor to participate in the Wiki by adding an article with that name. Later on, I also plan to handle searches, commenting and categories in a similar way.

Web.config

First off, we need to add a reference to the assemblies System.Web.Routing and System.Web.Abstractions. Then, add the UrlRouting module to web.config's httpModules list:

```
<add name="UrlRoutingModule" type="System.Web.Routing.UrlRoutingModule,System.Web.Routing,Version=3.5.(
/>
```

NOTE: If you want to use it in IIS7 Integrated Mode, you have to do another couple of web.config settings. Read more here: http://msdn.microsoft.com/en-us/library/cc668202.aspx
It doesn't matter if you add it before the registration of the UrlRewriteModule or after. But in my opinion it looks nicer after :-)

Creating a custom RouteHandler

Next thing, we need to create a custom route handler. A Route handler should implement IRouteHandler and contain a method that takes the requestContext and returns an IHTTPHandler - like - a Web Form :-) A Routehandler can transfer parameters from the url to the web form, by setting them in the HttpContext. In my case, I am passing the name of the new article as a parameter. Also, I set the CurrentPageLink property on the page I create to the current root page of the Wiki - so I can use it directly on the page. Since I only want users with permission to create new pages to be able to do this, I also perform an access check and return a 404 error if they don't have Create rights (one could argue that a security exception might also be valid here).

```
public class WikiRouteHandler : IRouteHandler{
    public PageReference WikiRoot { get; private set; }

    /// <summary>Constructor, set when RouteHandler is assigned to route./// </summary>
        ///                                                                 /// <param name="WikiRoot'
    {
        this.WikiRoot = WikiRoot;
    }

    public IHttpHandler GetHttpHandler(RequestContext requestContext)
    {
        //Check if user has create permissions to WikiRoot - if not, 404
                                                                    if (!DataFactory.Instance.GetP
            throw new HttpException(404, "Page Not Found");

        //Send parameters to NewPage
                                    foreach (var value in requestContext.RouteData.Values)
        {
```

```
            requestContext.HttpContext.Items[value.Key] = value.Value;
        }

        //Create a Page, based on the NewPage template to handle the new page
                                                                    var page = BuildManager.C
        page.CurrentPageLink = WikiRoot;

        return page as IHttpHandler;
    }
}
```

You can see a couple of other examples of custom route handlers here:

- http://msdn.microsoft.com/en-us/library/cc668202.aspx
- http://blogs.msdn.com/mikeormond/archive/2008/05/14/using-asp-net-routing-independent-of-mvc.aspx
- http://blogs.msdn.com/mikeormond/archive/2008/06/21/asp-net-routing-and-authorization.aspx

Setting up the routes

Last thing on my list: I need to set up the special url routes I want handled with my WikiRouteHandler. The ideal place to do this is in global.asax in the Application_Start - but since I want a more pluggable approach I turned to my own medicine and used a class inheriting from PluginAttribute to execute code at startup.

In the *static void start()* method there I simply registrered the routes like this:

```
RouteTable.Routes.Add("WikiLangRoute", new Route("{lang}/Wiki/{ArticleName}", new WikiRouteHandler(pr)
RouteTable.Routes.Add("WikiRoute", new Route("Wiki/{ArticleName}", new WikiRouteHandler(pr)));
```

In the first case the language and ArticleName is passed to the RouteHandler as parameters. Of course, this code assumes that the wiki is located in /Wiki/ - so naturally my next edit will be to change that dynamically - but anyway, I'm sure you get the idea.

Future potential

As I'm sure you can imagine this has a lot of potential uses ranging from special URL handling where it seems prudent, but perhaps it can also be used for a better way to provide multiple renderings for single page - based on the context or the url.

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