ALLAN THRAEN | 17 years ago | PDF |

EPISERVER PIE WITH FRESHLY CUT MVC

Or how to integrate ASP.NET MVC into an existing WebForms Web Application Project.

After having studied Scott Guthrie's excellent posts on ASP.NET MVC and attended a pretty good Tech Talk at Microsoft yesterday I decided the time was right to start playing around with integrating MVC code in the same project as a typical EPiServer CMS 5.1 web application. It turned out to be fairly easy - didn't take more than 1 hour of evening coding (there wasn't anything good on the telly anyway) - so here is a short recipe for all of you out there.

If you want to read more about ASP.NET MVC before you start, read here. To many it might present quite a paradigm change in web development (and I've personally not yet totally decided whether I think it's an improvement or not) so I imagine we'll be seeing a lot of cases where it's mixed into ordinary web projects, where it's strengths are best utilized. That's why it made sense to me to mix it with a standard EPiServer project.

Ingredients

- Working EPiServer CMS 5.1 web project with source and project files
- VS 2008
- ASP.NET 3.5 Extensions Preview (this recipe uses the December Preview - but if you're trying this after the MIX08 preview, drop a comment and let me know how it worked out!)

Step-by-step guide to baking your pie :-)

1. Open your EPiServer CMS project in VS 2008 and use the wizard to upgrade it. When it's upgraded, go to Project properties and set the target framework to ".NET Framework 3.5". Save and run in visual studio web server to make sure it works as expected.

2. In the project references, there is most likely a reference to System.Web.Extensions and System.Web.Wextensions.Design. Remove them, then add them again - but make sure you add the 3.6.0.0 version - not the 3.5.0.0 that you had before. This is actually the tricky part. Even though both assemblies are in the GAC I wasn't able to persuade VS to get the right version. If you also have problems, then copy the assembly out of the GAC and add it as a reference by browsing to the dll. It can be copied out of the GAC like this:
copy "C:\WINDOWS\assembly\GAC_MSIL\System.Web.Extensions\3.6.0.0__31bf3856ad364e35" "c:\myproject\"

3. Now it's time to update your web.config. First I changed the assembly binding to look like this:

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="System.Web.Extensions" publicKeyToken="31bf3856ad364e35"/>
    <bindingRedirect oldVersion="1.0.0.0-3.5.0.0" newVersion="3.6.0.0"/>
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="System.Web.Extensions.Design" publicKeyToken="31bf3856ad364e35"/>
    <bindingRedirect oldVersion="1.0.0.0-3.5.0.0" newVersion="3.6.0.0"/>
  </dependentAssembly>
</assemblyBinding>
```

4. Then it was time to setup the UrlRouting module that's a vital part of MVC. I found that it can coexist quite peacefully with EPiServer's Friendly Url rewriting - just add the module in the top of the http-modules list in web.config:

```
<add name="UrlRoutingModule"
     ="System.Web.Mvc.UrlRoutingModule,
     System.Web.Extensions, Version=3.6.0.0, Culture=neutral,
 type PublicKeyToken=31BF3856AD364E35"                        />
```

5. So far, so good. Now put it in the oven at 200 degrees and let it bake for 30 min and your EPiServer pie will be done. However we still need to sprinkle some MVC on top until now we've only prepared the platform - but we still need to make our fancy-smancy-state-of-the-art-super-microsoft-will-love-us-and-it-will-taste-great MVC code...And the first place I'll start is by setting up the URL Routing in Global.asax. First, make sure Global.asax.cs is included in your project and that Global.asax inherits from you code - that in turn inherits from EPiServer.Global.
Then, I put in the following code to ensure that everything in the url path /mvc/ is handled by the MVC Url Routing:

```
protected void Application_Start(Object sender, EventArgs e)
{
  RouteTable.Routes.Add(new Route
  {
    Url = "mvc/[controller]/[action]/[id]",
    Defaults = new { action = "Index", id = (int?)null },
```

```
    RouteHandler = typeof(MvcRouteHandler)
  });
}
```

6. Now, it's really just left for us to make some MVC functionality. As an example I decided to make two MVC Actions. One will get a list of all pages that are visible in menus and show it, the other will list all properties for a given page. That way we'll see some interaction with EPiServer as well. I made a "Controllers" folder in my project, along with a "Views" folder that had a subfolder "Page" (since I decided that my controller would be a PageController). Then I used the visual studio templates to generate a PageController, and two views. The data model for what I want to achieve already exists in EPiServer (PageData) so I saw no reason to make a new one.

7. In the PageController, I made two actions - one "Index" would perform a search for pages visible in menus and send them to a view, the other would show the properties of a page. Here's the code:

```
public class PageController : Controller
{

    [ControllerAction]
    public void Index()
    {
        PropertyCriteriaCollection pcc=new PropertyCriteriaCollection();
        pcc.Add(new PropertyCriteria() {
            Required = true,
            Condition = CompareCondition.Equal,
            Name = "PageVisibleInMenu",Value="True",
            Type=PropertyDataType.Boolean });
        PageDataCollection pdc = DataFactory.Instance.FindPagesWithCriteria(
            PageReference.RootPage, pcc);
        RenderView("Index",pdc);
    }

    [ControllerAction]
    public void ShowPageData(int id)
    {
        PageData p = DataFactory.Instance.GetPage(new PageReference(id));
        RenderView("PageDataView", p);
    }
}
```

8. Then it became time to look at the views. I modified the code-behind in each view to make it inherit from a generic ViewPage with a datatype corresponding to the data I was sending to it - like ViewPage<PageData>. From there on it was just a matter of spraying a few simple lines of mixed html and code in the view - and voila!

```
<ul>
 <% foreach (EPiServer.Core.PageData p in ViewData)
    { %>
 <li><%= Html.ActionLink(p.PageName,new {controller="Page", action="ShowPageData", id=p.PageLink.ID
 <% } %>
 </ul>
```

9. Follow the same approach and you'll easily be able to display the properties as well.

Enjoy!

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