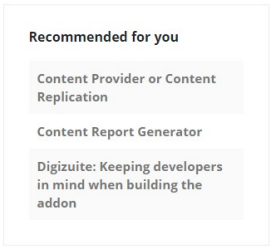
ALLAN THRAEN | 5 years ago | PDF |

Optimizely (Episerver) AI CMS

EPISERVER ADVANCE RECOMMENDATIONS ON CODEART

Some times you have so much great content on your website that you just wish you had a librarian to let your visitors know what to read next. And with Episerver Advance (Content Recommendations) you can at least have something that comes pretty close. I have been lucky enough to try it out on my site.

There are many approaches to doing content recommendations - and I'll probably do a number of blog posts on this topic as content+AI is one of my favorite topics. In the past I've done it using search fairly efficiently, I've done it using datamined Google Analytics (a great approach I might share in a future post) but of course there is also the Ferrari-approach, using all the sophiscated AI that Episerver has put together and made easily available. And since I got sponsored the service I did just that. It's been online for a few months and you might have noticed it on the bottom of the sidebar for each blog post. It looks something like this:



With Episerver tracking and Insight configured, using Advance is pretty straight forward. The hard part is really to keep track of all the different connection keys you get for all these different services :-)

The sample code provided on github has an approach I found a bit strange at first. It's a regular block, with a model, a controller and a view - just as we know it - but the view then contains javascript calling back to a Web API Controller that in turn calls the recommendation service, fetches results, renders them and returns the html to be inserted. So basically an AJAX approach.

But after some experimentation I must admit that in this case it's probably the best approach as a slow external service won't cause the site to slow down or crash - as it's loaded afterwards and async.

So, I chose the same solution, but simplified a few places. I also added a view for the final output so I can easily update the look & feel should I suddenly get an urge to show of my design skills.

I won't bother you with the controller or model for the base block, as that's completely straightforward (yes, it could even be using the default controller). The view however that does the callback looks like this in my code:

```
@using EPiServer.Core
@using EPiServer.Web.Mvc.Html
@model AllanTech.Web.Models.ViewModels.AdvanceViewModel
@{ Layout = null;}

<div class="widget categories">
    <header>
        <h3 class="h6" @Html.EditAttributes(m => m.CurrentBlock.Name)>@Model.CurrentBlock.Name</h3>
    </header>
    <div class="advancerecs"></div>

                    $(function(){
                        $.ajax('/Advance/?contentId=@Model.ContentId&numRecs=@Model.CurrentBlock.Numbe
                            success:function (data) {
                                $('.advancerecs').html(data);
                            }
                        });
                    });
    <script>
</div>
```

That, in turn, calls back to my recommendations controller which fetches recommendations from Advance:

```
    public class AdvanceRecommendationsController : Controller
    {
        private readonly IRecommendationService _recommendationService;

        public AdvanceRecommendationsController(IRecommendationService recommendationService)
                                                                                            {
            _recommendationService = recommendationService;
        }
        // GET: AdvanceRecommendations
        [Route("advance")]
        public async Task<ActionResult> Index(string contentId, int numRecs=3)
```

```
                                                                    {
            var rRequest = new RecommendationRequest
            {
                siteId = SiteDefinition.Current.Id.ToString(),
                context = new Context { contentId = contentId, languageId = "en" },
                numberOfRecommendations = numRecs
            };
            var res = await _recommendationService.Get(this.HttpContext, rRequest);
            return View(res);
        }
    }
```

And this is then rendered and returned to the calling javascript through this very basic view:

```
@using EPiServer.Core
@using EPiServer.Web.Mvc.Html
@model IEnumerable<EPiServer.Personalization.CMS.Model.RecommendationResult>
@{ Layout = null;}
@foreach (var m in Model)
{
    <div class="item d-flex justify-content-between">@Html.ContentLink(m.Content.ContentLink)</div>
}
```

As you can see - it's pretty straight forward.

Cool, but does it actually keep visitors engaged on the site longer?

Well, I wish I had a clear statistic that indicated that since I introduced this feature the number of pages per visit has increased tremendously - but sadly I don't. I'm working on improving my analytics setup though to give better insights as to how often the Advance links keep visitors on the site, but the data isn't ready yet.

But try to check it out when you are browsing my blog and see if those blog posts match your interests. Drop a comment and let me know what you think.

Optimizely (Episerver) AI CMS

RECENT POSTS

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