ALLAN THRAEN | 1 years ago | PDF |

Tips and Tricks C# Personalization Optimizely (Episerver)

PREVIEW MULTIPLE VISITOR GROUPS DIRECTLY WHILE BROWSING YOUR OPTIMIZELY SITE

Visitor groups are great - it's an easy way to add personalization towards market segments to your site. But it does come with it's own set of challenges if used intensively. For example it can be hard to predict how any given page will look for visitors with a specific combination of visitor groups - and viewing it in a proper way often requires more than what you see in the quick preview mode. Here's a bit of code that will help you out.

Around 12 years ago, in the groundbreaking release of Episerver CMS 6 R2 (which I will still argue was probably the best release ever due to sublime product management) a new feature saw the light of day: Visitor Groups.

It provided editors with the much needed capability of recognizing segments of visitors on their site - and target specific pieces of content for them on the site. However, with great power comes great responsibility - and the caveat with visitor groups is that they can be used to solve so many different problems on so many different dimensions that you risk ending up with too many groups to manage - and an amount of usages that's impossible to fathom.

I seem to recall some discussions in the product/development team a long time ago concerning if we should prioritize UI for handling abundant amounts of visitor groups in v1 or wait until the next version - and someone said something along the lines of "Surely no-one will ever need more than just a handful of visitor groups" - a quote that today makes me think of the famous line: "640K ought to be enough for anybody."

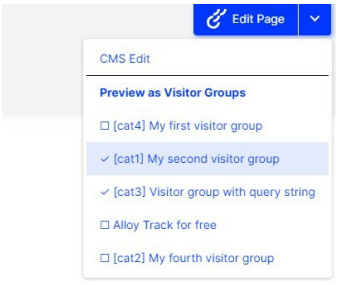
Anyway, a common problem I hear from many Episerver/Optimizely CMS clients is that they have gone 'all-in' on visitor groups, but then started running into problems.

Typical problems include:

- Managing the visitor groups when there are many (something I will address in a future blog post)
- Getting an overview of where they are used (for which you can use this: https://www.codeart.dk/blog/2018/10/cms-audit-update---visitor-groups-included/)
- Experiencing what a visitor with a certain combination of visitor groups will see for a given page.

Now, the best approach might be to rethink how you use visitor groups and reduce the dimensions until it's comprehendable. But as usual we don't have time for rational solutions. So - I've come up with this little tool - code available in the GIST below, for CMS 12.

It essentially extends the QuickNavigationMenu that you see when you are logged in as an editor with a list of the visitor groups in use on the current page (or in any blocks used on it somewhere) - and gives you the options of viewing it as one - or many of those groups, simply by selecting it and "checking the box".



It's surprisingly easy to extend the QuickNavigationMenu - you just have to implement an IQuickNavigatorItemProvider and register it with dependency injection. To make it work smoothly I also included a couple of extension methods to recursively find content used on a page - and identify the visitor groups used in a content element.

I plan to share a work-in-progress soon with an improved Visitor Groups Manager that can help you manage the existing visitor groups - who knows, maybe I'll even combine it with this to make a VisitorGroupsEnhancement Add-on. Let me know your thoughts in the comments below!

```
1  public class Startup(
2
3  //...
```

```
    public void ConfigureServices(IServiceCollection services)
    {
      //...

      services.AddSingleton<IQuickNavigatorItemProvider, VisitorGroupQuickNavigationProvider>();

      //...
    }
  }
```

StartUp.cs hosted with ♥ by GitHub view raw

```
using EPiServer;
using EPiServer.Core;
using EPiServer.Personalization.VisitorGroups;
using EPiServer.ServiceLocation;
using EPiServer.Shell.Profile.Internal;
using EPiServer.Web;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Linq;

namespace CodeArt.Optimizely.VisitorGroupManager
{
  /// <summary>
  /// Provides a quick navigation to impersonate the combination of visitor groups used by a page.
  /// </summary>
  public class VisitorGroupQuickNavigationProvider : IQuickNavigatorItemProvider
  {
    private readonly IContentRepository _contentRepo;
    private readonly IHttpContextAccessor _httpContextAccessor;
    private readonly ICurrentUiCulture _currentUiCulture;
    private readonly IVisitorGroupRepository _vgRepo;

    public VisitorGroupQuickNavigationProvider(
      IContentRepository contentRepo,
      IHttpContextAccessor httpContextAccessor,
      ICurrentUiCulture currentUiCulture,
      IVisitorGroupRepository vgRepo)
    {
      this._contentRepo = contentRepo;
      this._httpContextAccessor = httpContextAccessor;
      this._currentUiCulture = currentUiCulture;
      this._vgRepo = vgRepo;
    }
    public int SortOrder => 100;

    internal Injected<UIPathResolver> UIUriResolver { get; set; }

    private bool IsPageData(ContentReference currentContentLink) => this._contentRepo.Get<IContent>(currentContentLink) is PageData;

    public IDictionary<string, QuickNavigatorMenuItem> GetMenuItems(ContentReference currentContent)
    {
      IContent c = _contentRepo.Get<IContent>(currentContent);
      var referencedContent = _contentRepo.FetchReferencedContentRecursively(c).ToList();
      var groups = referencedContent.SelectMany(rc => _vgRepo.ExtractVisitorGroups(rc)).Distinct().ToList();

      QuickNavigatorMenu quickNavigatorMenu = new QuickNavigatorMenu(this._httpContextAccessor, this._currentUiCulture);
      quickNavigatorMenu.CurrentContentLink = !(currentContent != (ContentReference)null) || !this.IsPageData(currentContent) ? (ContentReference)ContentRe

      var active = this._httpContextAccessor.HttpContext.Request.Query["visitorgroupsByID"].ToString().Split('|').Where(s => s!= string.Empty).Select(Guid.Parse).

      string firstDivider = "\" style='border-top: 1px solid black;font-weight:bold' t=\"";

      quickNavigatorMenu.Add("divider", new QuickNavigatorMenuItem("Preview as Visitor Groups", "#" + firstDivider, (string)null, "false", (string)null));

      //List visitor groups on this page
      foreach (var g in groups)
      {
        //Create link
        var url = new UrlBuilder(_httpContextAccessor.HttpContext.Request.Path.ToString()+_httpContextAccessor.HttpContext.Request.QueryString.ToString())

        if (active.Contains(g.Id))
        {
          var newActive=active.Except(new Guid[] { g.Id }).ToArray();
          if(newActive.Any()) url.QueryCollection["visitorgroupsByID"] = string.Join("|", newActive);
          else url.QueryCollection.Remove("visitorgroupsByID");
          quickNavigatorMenu.Add(g.Id.ToString(), new QuickNavigatorMenuItem("&check; " + g.Name, url.ToString(), (string)null, "true", (string)null));
        } else
        {
          var newActive = active.Union(new Guid[] { g.Id }).ToArray();
          url.QueryCollection["visitorgroupsByID"]=string.Join("|", newActive);
          quickNavigatorMenu.Add(g.Id.ToString(), new QuickNavigatorMenuItem("&#9744; " + g.Name, url.ToString(), (string)null, "true", (string)null));
        }
      }

      return quickNavigatorMenu.Items;
    }
  }
}
```

VisitorGroupQuickNavigationProvider.cs hosted with ♥ by GitHub view raw

```
using EPiServer;
using EPiServer.Core;
using EPiServer.Personalization.VisitorGroups;
using System;
using System.Collections.Generic;
using System.Linq;

namespace CodeArt.Optimizely.VisitorGroupManager
{
  public static class VisitorGroupsHelper
  {
    /// <summary>
    /// Analyzes which visitor groups are used in a piece of content and returns a list of those groups.
    /// </summary>
    /// <param name="vgRepo">The VisitorGroupRepository to extend</param>
    /// <param name="content">The content to analyze</param>
    /// <returns></returns>
    public static IEnumerable<VisitorGroup> ExtractVisitorGroups(this IVisitorGroupRepository vgRepo,IContent content)
    {
      foreach (var p in content.Property)
      {
        if (p.Value == null) continue;
        if (p.PropertyValueType == typeof(ContentArea))
        {
          var ca = p.Value as ContentArea;
          if (ca == null) continue;
          foreach (var f in ca.Items.Where(i => i.AllowedRoles != null && i.AllowedRoles.Any()))
```

```
            {
                //Match! This page uses the visitor groups in f.AllowedRoles. Record.
                foreach (var r in f.AllowedRoles)
                {
                    yield return vgRepo.Load(Guid.Parse(r));
                }
            }
        }
        else if (p.PropertyValueType == typeof(XhtmlString))
        {
            var ca = p.Value as XhtmlString;
            if (ca == null) continue;
            foreach (var f in ca.Fragments.Where(fr => fr is EPiServer.Core.Html.StringParsing.PersonalizedContentFragment))
            {

                var j = f as EPiServer.Core.Html.StringParsing.PersonalizedContentFragment;
                var roles = j.GetRoles();
                foreach (var r in roles)
                {
                    yield return vgRepo.Load(Guid.Parse(r));
                }
            }
        }
    }
}

/// <summary>
/// Fetches content used by a piece of content (in ContentAreas or XhtmlStrings), including the content itself.
/// </summary>
/// <param name="repo">The IContentRepository to extend on</param>
/// <param name="content">The original content</param>
/// <returns>Enumeration of IContent referenced</returns>
public static IEnumerable<IContent> FetchReferencedContentRecursively(this IContentRepository repo, IContent content)
{
    yield return content;
    foreach (var p in content.Property)
    {
        if (p.Value == null) continue;
        if (p.PropertyValueType == typeof(ContentArea))
        {
            var ca = p.Value as ContentArea;
            if (ca == null) continue;
            foreach (var f in ca.Items)
            {
                foreach (var y in repo.FetchReferencedContentRecursively(repo.Get<IContent>(f.ContentLink)))
                    yield return y;
            }
        }
        else if (p.PropertyValueType == typeof(XhtmlString))
        {
            var ca = p.Value as XhtmlString;
            if (ca == null) continue;
            foreach (var f in ca.Fragments.Where(fr => fr is EPiServer.Core.Html.StringParsing.ContentFragment))
            {
                var j = f as EPiServer.Core.Html.StringParsing.ContentFragment;
                foreach (var y in repo.FetchReferencedContentRecursively(repo.Get<IContent>(j.ContentLink)))
                    yield return y;
            }
        }
    }

}
```

VisitorGroupsHelper.cs hosted with ♥ by GitHub view raw

Tips and Tricks C# Personalization Optimizely (Episerver)

RECENT POSTS

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