ALLAN THRAEN | 5 years ago | PDF |

Tips and Tricks CMS Addon Development Optimizely (Episerver)

CONTENT REPORT GENERATOR

I helped a client with a cool little report generator that can give them an easy overview of all their content - and related metadata, that can be opened in excel and easily sorted, filtered and aggregated. Here it is.

When you have a lot of content, produced by many editors - and especially across many sites, pages and blocks - it's quite important to get a good overview over your content to answer some essential questions like:

- Who is producing all the content
- Where is the old content
- How are my review and approval rules implemented and used
- What is outdated and should be removed
- Which content needs to be translated
- And many other kinds of content governance.

Of course there is the good old Episerver Report center. And thanks to this great example by Henrik Fransas, it's pretty easy to figure out how to build your own reports and export them. But since those reports are generated on the fly (in the request) you can end up waiting for that download for a long time - if you don't run into a time out. Sure, not a problem when you have a few hundred content items - but not so fun when you are dealing with many hundred thousands (which is when you really need the overview). So, for this report I took the approach of letting a scheduled job generate the report and then make it available for download.

Also, most of the existing reports in the report center are pretty much focused purely on the content in the content repository. But these days there is quite a lot more interesting meta-information around the content. Who has access to it, how many versions does it have, where is it used, who is reviewing it and so on - all stuff that could help you achieve better content governance.

In the past I've done prototypes, examples, demos and experiments using PowerBI for this sort of task, but in this case an excel file is fine to keep people happy most of the time, so why complicate things.

Here is the initial code I came up with - feel free to use and modify it to fit your reporting needs:

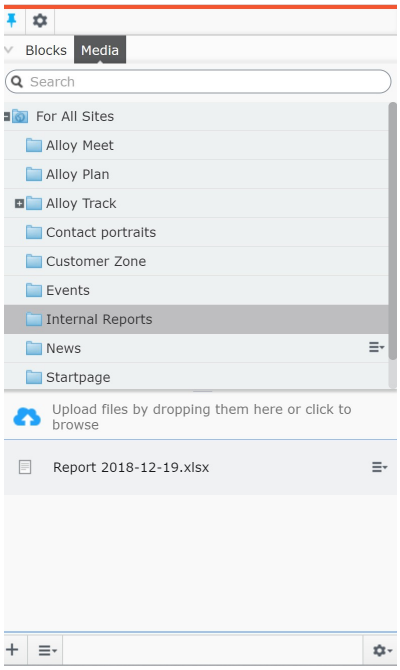
```
using System;
using EPiServer;
using EPiServer.Core;
using EPiServer.PlugIn;
using EPiServer.Scheduler;
using EPiServer.ServiceLocation;
using System.Linq;
using System.Collections.Generic;
using EPiServer.Web;
using EPiServer.DataAbstraction;
using OfficeOpenXml;
using System.Web.Mvc;
using EPiServer.Web.Mvc.Html;
using EPiServer.Approvals;
using EPiServer.Approvals.ContentApprovals;
using MyAlloy.Models.Media;
using EPiServer.Framework.Blobs;
using System.IO;

namespace CodeArt.Examples.Addons.Jobs
{
    // REMEMBER: To add EPPlus through nuget to the project!
    // https://github.com/JanKallman/EPPlus

    [ScheduledPlugIn(DisplayName = "Content Report Generator")]
    public class ContentReportJob : ScheduledJobBase
    {
        private bool _stopSignaled;

        public ContentReportJob()
        {
            IsStoppable = true;
        }

        /// <summary>
        /// Called when a user clicks on Stop for a manually started job, or when ASP.NET shuts down.
        /// </summary>
        public override void Stop()
        {
            _stopSignaled = true;
        }

        /// <summary>
        /// Called when a scheduled job executes
        /// </summary>
        /// <returns>A status message to be stored in the database log and visible from admin mode</returns>
        public override string Execute()
        {
            //Call OnStatusChanged to periodically notify progress of job for manually started jobs
            OnStatusChanged("Starting to build report");
            int cnt = 0;


            //Add implementation
            var repo = ServiceLocator.Current.GetInstance<IContentRepository>();
            var vrepo = ServiceLocator.Current.GetInstance<IContentVersionRepository>();
            var sites = ServiceLocator.Current.GetInstance<ISiteDefinitionRepository>();
```

```
var trepo = ServiceLocator.Current.GetInstance<IContentTypeRepository>();
var resolver = ServiceLocator.Current.GetInstance<ISiteDefinitionResolver>();
var apdefrepo = ServiceLocator.Current.GetInstance<IApprovalDefinitionRepository>();

var url = new UrlHelper();
int trashid = trepo.Load("SysRecycleBin").ID;

//Ensure output folder
ContentReference outputRef = ContentReference.EmptyReference;
ContentFolder outputfolder=repo.GetChildren<ContentFolder>(ContentReference.GlobalBlockFolder).Where(cf => cf.Name=="Internal Reports").FirstOrDefaul
if (outputfolder == null)
{
    var tmp = repo.GetDefault<ContentFolder>(ContentReference.GlobalBlockFolder);
    tmp.Name = "Internal Reports";
    //TODO: Access rights, only admin access!!
    outputRef = repo.Save(tmp, EPiServer.DataAccess.SaveAction.Publish,EPiServer.Security.AccessLevel.NoAccess);
}
else outputRef = outputfolder.ContentLink;

//Fetch all IContent (not in trash), build report
Queue<ContentReference> todo = new Queue<ContentReference>();
Dictionary<ContentReference, bool> done = new Dictionary<ContentReference, bool>();
todo.Enqueue(ContentReference.GlobalBlockFolder);
todo.Enqueue(ContentReference.SiteBlockFolder);
sites.List().ToList().ForEach(sd => {
    todo.Enqueue(sd.ContentAssetsRoot);
    todo.Enqueue(sd.GlobalAssetsRoot);
    todo.Enqueue(sd.SiteAssetsRoot);
    todo.Enqueue(sd.StartPage);
});
using (var package = new ExcelPackage())
{
    ExcelWorksheet ws = package.Workbook.Worksheets.Add("Content");
    //Header row
    ws.Cells[1, 1].Value = "Content ID";
    ws.Cells[1, 2].Value = "Type";
    ws.Cells[1, 3].Value = "Main Type";
    ws.Cells[1, 4].Value = "Url";
    ws.Cells[1, 5].Value = "Site Name";
    ws.Cells[1, 6].Value = "Breadcrumb";
    ws.Cells[1, 7].Value = "Name";
    ws.Cells[1, 8].Value = "Status of latest draft";
    ws.Cells[1, 9].Value = "Created By";
    ws.Cells[1, 10].Value = "Created Date";
    ws.Cells[1, 11].Value = "Changed By";
    ws.Cells[1, 12].Value = "Changed Date";
    ws.Cells[1, 13].Value = "Saved By";
    ws.Cells[1, 14].Value = "Saved Date";
    ws.Cells[1, 15].Value = "Published Date";
    ws.Cells[1, 16].Value = "Languages";
    ws.Cells[1, 17].Value = "Uses Count";
    ws.Cells[1, 18].Value = "Uses";
    ws.Cells[1, 19].Value = "Versions";
    ws.Cells[1, 20].Value = "Reviewers";

    int row = 2;
    while (todo.Count > 0)
    {
        var cref = todo.Dequeue();
        if (done.ContainsKey(cref)) continue; //Skip if we already did this
        done.Add(cref, true);
        var mastercnt = repo.Get<IContent>(cref);
        if (mastercnt.ContentTypeID == trashid) continue; //Skip if it's the recycle bin
        repo.GetChildren<IContent>(cref).Select(ch => ch.ContentLink).ToList().ForEach(ch => todo.Enqueue(ch)); //Add children to todo-list
                                                                                                                //Get ancestors
        var ancestors = repo.GetAncestors(cref).Select(a => a.Name).ToList();
        //Get versions
        var versions = vrepo.List(cref);
        //Which version should we use?
        var draft = versions.Where(cv => cv.IsMasterLanguageBranch).OrderByDescending(cv => cv.MasterVersionID).First();
        var draftcnt = repo.Get<IContent>(draft.ContentLink);
        var refs = repo.GetReferencesToContent(cref, false); //Does this work?
        var site = resolver.GetByContent(cref, true);

        var apdef = apdefrepo.ResolveAsync(cref).Result;
        var revs = apdef?.Definition?.Steps?.SelectMany(st => st.Reviewers).Select(r => r.Name).ToArray();
        string reviewers = (revs == null || apdef.Definition.IsEnabled == false) ? "(No Approval Flow)" : string.Join(",", revs);

        ws.Cells[row, 1].Value = cref.ID;
        ws.Cells[row, 2].Value = trepo.Load(mastercnt.ContentTypeID).Name;
        ws.Cells[row, 3].Value = (mastercnt is PageData) ? "Page" : (mastercnt is MediaData) ? "Media" : (mastercnt is BlockData) ? "Block" : "Other";
        ws.Cells[row, 4].Value = (mastercnt is PageData || mastercnt is MediaData) ? url.ContentUrl(cref): string.Empty;
        ws.Cells[row, 5].Value = site?.Name;
        ws.Cells[row, 6].Value = string.Join(" > ",ancestors.ToArray());
        ws.Cells[row, 7].Value = draftcnt.Name;
        ws.Cells[row, 8].Value = draft.Status.ToString();
        ws.Cells[row, 9].Value = (draftcnt as IChangeTrackable)?.CreatedBy;
        ws.Cells[row, 10].Value = (draftcnt as IChangeTrackable)?.Created.ToString("yyyy-MM-dd HH:mm");
        ws.Cells[row, 11].Value = (draftcnt as IChangeTrackable)?.ChangedBy; ;
        ws.Cells[row, 12].Value = (draftcnt as IChangeTrackable)?.Changed.ToString("yyyy-MM-dd HH:mm");
        ws.Cells[row, 13].Value = draft.SavedBy;
        ws.Cells[row, 14].Value = draft.Saved.ToString("yyyy-MM-dd HH:mm");
        if (mastercnt is IVersionable)
        {
            ws.Cells[row, 15].Value = ((mastercnt as IVersionable).StartPublish.HasValue) ? (mastercnt as IVersionable).StartPublish.Value.ToString("yyyy-MM-dd
        }
        if (mastercnt is ILocalizable)
        {
            ws.Cells[row, 16].Value = (mastercnt is ILocalizable) ? string.Join(", ", (mastercnt as ILocalizable).ExistingLanguages.Select(ci => ci.TwoLetterISOLan
        }
        ws.Cells[row, 17].Value = refs.Count().ToString();
        ws.Cells[row, 18].Value = string.Join(", ", refs.Select(r => r.OwnerName + "("+r.OwnerID.ToString()+")").ToArray());
        ws.Cells[row, 19].Value = versions.Count().ToString();
        ws.Cells[row, 20].Value = reviewers;

        //For long running jobs periodically check if stop is signaled and if so stop execution
        if (_stopSignaled)
        {
            break; //Still finish report
        }
        row++;
        cnt++;
        OnStatusChanged($"In progress, done with {cnt} content items, {todo.Count} waiting.");
    }
    //Save excel file
    string repname= "Report " + DateTime.Now.ToString("yyyy-MM-dd") + ".xlsx";
    GenericMedia reportcnt = repo.GetChildren<GenericMedia>(outputRef).Where(ch => ch.Name == repname).FirstOrDefault();
    if (reportcnt != null)
    {
        reportcnt = (GenericMedia) reportcnt.CreateWritableClone();
    }
    else
    {
```

```
                reportcnt = repo.GetDefault<GenericMedia>(outputRef);
                reportcnt.Name = repname;
            }
            var blobfactory= ServiceLocator.Current.GetInstance<IBlobFactory>();
            var blob=blobfactory.CreateBlob(reportcnt.BinaryDataContainer, ".xlsx");
            using (var s = blob.OpenWrite())
            {
                BinaryWriter w = new BinaryWriter(s);
                w.Write(package.GetAsByteArray());
                w.Flush();
            }
            reportcnt.BinaryData = blob;
            repo.Save(reportcnt, EPiServer.DataAccess.SaveAction.Publish, EPiServer.Security.AccessLevel.NoAccess);

        }
        return $"Job done. Wrote {cnt} content items";
    }
  }
}
```

ContentReportJob.cs hosted with ♥ by GitHub view raw

It will generate a report as a media file in Episerver, in an automatically created folder called "Internal Reports". I advice that you after the initial creation set access rights on the folder to ensure that the reports can only be accessed by specific roles!



For the report I included a bit of the code used in my previous post for gathering information about who is configured to review the content - and a bunch of other code to get a list of places that use the content, and the versions of the content.

To generate the excel format document I use the EPPlus library, easily added to your project through nuget.

A fairly easy modification I would recommend might be to generate plain CSV files instead (I ended up doing that for the client, but haven't updated the example code yet). Using EPPlus you build up the excel file in memory and write it in the end - which I expect could be a problem if you excel files turns really big. CSV files you can generate, write and flush to disk line-by-line, hence eliminating memory issues.

Here you can find an example Report from a standard Alloy site:

ReportsExample.xlsx

Tips and Tricks | CMS | Addon Development | Optimizely (Episerver)

RECENT POSTS

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