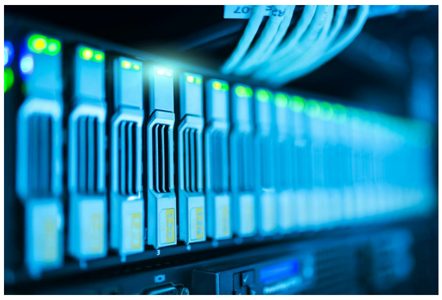
ALLAN THRAEN | 5 years ago | PDF |

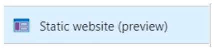
Azure Optimizely (Episerver)

EPISERVER STATIC WEB SITE GENERATOR



Azure Storage has a new cool feature in preview - Static Website. But what exactly does it do - and how can I connect my Episerver installation to it? I decided to find out.

Last night, I was starting up a new instance of Azure Storage, and a new feature caught my eye: Static Website (preview).



Naturally, I couldn't help myself and had to have a closer look. On various occassions I've served both html, javascript and images directly from Azure storage, and I knew that it's possible to attach your own domain name to a storage, so what exactly did this feature add in order to have a full static website on Azure?

When you enable static websites on your storage account, a new web service endpoint is created of the form <account.name>.<zone-name>.web.core.windows.net. The web service endpoint always allows anonymous read access, returns formatted HTML pages in response to service errors, and allows only object read operations. The web service endpoint returns the index document in the requested directory for both the root and all subdirectories. When the storage service returns a 404 error, the web endpoint returns a custom error document if you configured it.

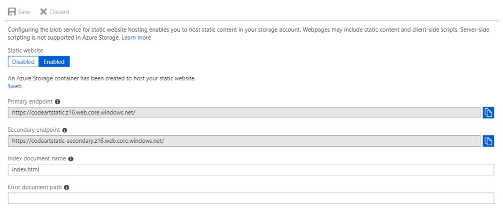
So - looks like we're basically getting a properly configured wwwroot to our website (albeit the container name is $web). That's pretty cool! I guess I haven't been this excited about static websites since 1996 :-)

Now, if only I had a static website to put up there, I definitely would. And there - I ended up in a nostalgia trip for a brief while, remember how I saved up the money from my newspaper route for a license for MS FrontPage and built my first round of static website. Remember the <BLINK> tag? Oh well. I digress.

Leaving the nostalgia for a moment, static websites are probably underrated today. Many websites today are in a sense static - at least to the point where they don't need much server side processing to deal with their visitors - it's only when it comes to editing and content management that the 'dynamic' part kicks in.

Now, this got me asking the question I somehow always find myself asking: "Can I connect Episerver to this" -and as usual of course the answer is 'yes'. In fact, I remember back in CMS 4/5/6 where we had a mirroring functionality that would in fact mirror your site to static files.

Anyway - I enabled the functionality, connected my own subdomain and set out to connect an Alloy site on Episerver CMS to Azure Storage Static Websites.



Configuring the blob service for static website hosting enables you to host static content in your storage account. Webpages may include static content and client-side scripts. Server-side scripting is not supported in Azure Storage. Learn more

Static website: Disabled [✓ Enabled]

An Azure Storage container has been created to host your static website.
$web

Primary endpoint: https://codeartstatic.z16.web.core.windows.net/

Secondary endpoint: https://codeartstatic-secondary.z16.web.core.windows.net/

Index document name: index.html

Error document path:

Since this is just a quick prototype, I decided on doing a scheduled job. They are fast and easy to build, can both be run manually and on a schedule and works like a charm. A slightly better implementation would of course also list to content events, so it could instantly update the static site whenever an editor made a change.

I put my connectionstring to Azure storage in my web.config and started coding using the visual studio template for Scheduled Jobs. First order of business is of course to initialize the connection to the blob storage:

```
            //Configure Blog storage
            account = CloudStorageAccount.Parse(WebConfigurationManager.AppSettings["StaticStorage"])
```

```
        container = account.CreateCloudBlobClient().GetContainerReference("$web");
```

I also wrote a few helper methods - here is the one that gets static versions of the content and uploads it to storage:

```
        protected int TraverseSite(ContentReference n, string language)
                                                                {
            int cnt = 0;
            var u = UrlResolver.Current.GetUrl(n,language);
            //Url is null if it's not url adressable (for example block or folder)
            if (u != null)
            {
                var uri = new Uri(u);
                var rel = uri.AbsolutePath;
                OnStatusChanged(String.Format("Fetching {0}", rel));
                try
                {
                    WebClient wc = new WebClient();
                    var data = wc.DownloadData(u);
                    var name = rel.TrimStart('/');
                    if (name.EndsWith("/")) name = name + DEFAULTFILENAME;
                    var blob = container.GetBlockBlobReference(name);
                    blob.Properties.ContentType = wc.ResponseHeaders[HttpResponseHeader.ContentType];
                    blob.Properties.ContentEncoding = wc.ResponseHeaders[HttpResponseHeader.ContentEnc
                    blob.Properties.CacheControl = wc.ResponseHeaders[HttpResponseHeader.CacheControl]
                    blob.UploadFromByteArray(data, 0, data.Length);
                    blob.SetProperties();
                    cnt++;
                }
                catch
                {
                    //TODO: Log error
                }
            }
            //Get Content Assets recursively
            var l = _assethelper.GetAssetFolder(n);
            if (l != null)
            {
                foreach (var a in _loader.GetDescendents(l.ContentLink))
                {
                    cnt += TraverseSite(a,language);
                }
            }
            return cnt;
         }
```

There are several ways to approach getting generated content. In this case I took the easy way, bound to work - which is to simply fetch it as an anonymous user using a webclient. That way I didn't have to worry about access control, publish status and so on. Also, I could simply read the response parameters and set them against the blob parameters (this is important, as otherwise blobstorage will not serve the html, instead, just send the html file out as an attachment).

You can see the full code in the GIST below.

Then, all that was left to do was to run the scheduled job.

Obviously, some features won't work. Like the search. And I haven't handled old-style permanent links, so if there are any that's just a shame. And it might not even be all the useful - I mean - if you're already running Episerver CMS, why would you want to go static? Well - I think there can be some use-cases, although they might be more theoretical.

Although I'm not considering license cost, etc. it's worth pointing out that Azure storage costs next to nothing, is fast, reliable and very easy to configure geo-redundant. Turning on Azure CDN is also a simple configuration change. Food for thought.

Learn more about the static websites of Azure storage here:
https://azure.microsoft.com/en-us/blog/azure-storage-static-web-hosting-public-preview/

```
using System;
using EPiServer.Core;
using EPiServer.PlugIn;
using EPiServer.Scheduler;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;
using EPiServer;
using EPiServer.ServiceLocation;
using EPiServer.Web.Routing;
using System.Net;
using System.Web.Hosting;
using System.IO;
using System.Web;
using System.Collections.Generic;
using EPiServer.DataAbstraction;
using System.Web.Configuration;

namespace StaticAlloy.StaticSiteGenerator
{
  [ScheduledPlugIn(DisplayName = "Generate Static Site")]
  public class StaticGeneratorJob : ScheduledJobBase
  {
    public const string DEFAULTFILENAME = "index.html";
    private bool _stopSignaled;

    /// <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;
    }

    public StaticGeneratorJob(IContentLoader loader, ContentAssetHelper assethelper, ILanguageBranchRepository languagerepo)
    {
      _loader = loader;
      _assethelper = assethelper;
      _languagerepo = languagerepo;
      IsStoppable = true;
    }

```

```
        protected CloudStorageAccount account;
        protected CloudBlobContainer container;
        protected IContentLoader _loader;
        protected ContentAssetHelper _assethelper;
        protected ILanguageBranchRepository _languagerepo;

        protected int TraverseSite(ContentReference n, string language)
        {
            int cnt = 0;
            var u = UrlResolver.Current.GetUrl(n,language);
            //Url is null if it's not url adressable (for example block or folder)
            if (u != null)
            {
                var uri = new Uri(u);
                var rel = uri.AbsolutePath;
                OnStatusChanged(String.Format("Fetching {0}", rel));
                try
                {
                    WebClient wc = new WebClient();
                    var data = wc.DownloadData(u);
                    var name = rel.TrimStart('/');
                    if (name.EndsWith("/")) name = name + DEFAULTFILENAME;
                    var blob = container.GetBlockBlobReference(name);
                    blob.Properties.ContentType = wc.ResponseHeaders[HttpResponseHeader.ContentType];
                    blob.Properties.ContentEncoding = wc.ResponseHeaders[HttpResponseHeader.ContentEncoding];
                    blob.Properties.CacheControl = wc.ResponseHeaders[HttpResponseHeader.CacheControl];
                    blob.UploadFromByteArray(data, 0, data.Length);
                    blob.SetProperties();
                    cnt++;
                }
                catch
                {
                    //TODO: Log error
                }
            }
            //Get Content Assets recursively
            var l = _assethelper.GetAssetFolder(n);
            if (l != null)
            {
                foreach (var a in _loader.GetDescendents(l.ContentLink))
                {
                    cnt += TraverseSite(a,language);
                }
            }
            return cnt;
        }

        public static string[] GetFiles(string path, string searchPattern, SearchOption searchOption)
        {
            string[] searchPatterns = searchPattern.Split('|');
            List<string> files = new List<string>();
            foreach (string sp in searchPatterns)
                files.AddRange(System.IO.Directory.GetFiles(path, sp, searchOption));
            files.Sort();
            return files.ToArray();
        }

        public int TraverseFiles(string basefolder,string folder, string pattern, bool recursive)
        {
            int cnt = 0;
            foreach(var f in GetFiles(Path.Combine(basefolder,folder), pattern, (recursive)? SearchOption.AllDirectories:SearchOption.TopDirectoryOnly))
            {
                string rel = f.Replace(basefolder, "");
                OnStatusChanged(String.Format("Uploading {0}", rel));
                var blob=container.GetBlockBlobReference(rel);
                var mime=MimeMapping.GetMimeMapping(Path.GetFileName(f));
                blob.Properties.ContentType = mime;
                blob.UploadFromFile(f);
                blob.SetProperties();
                cnt++;
            }
            return cnt;
        }

        /// <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(String.Format("Starting execution of {0}", this.GetType()));

            //Configure Blog storage
            account = CloudStorageAccount.Parse(WebConfigurationManager.AppSettings["StaticStorage"]);
            container = account.CreateCloudBlobClient().GetContainerReference("$web");

            //Traverse content
            int cnt = 0;
            foreach (var b in _languagerepo.ListEnabled())
            {
                List<ContentReference> lst = new List<ContentReference>();
                lst.Add(ContentReference.StartPage);
                lst.AddRange(_loader.GetDescendents(ContentReference.StartPage));
                lst.Add(ContentReference.SiteBlockFolder);
                lst.AddRange(_loader.GetDescendents(ContentReference.SiteBlockFolder));
                foreach (var n in lst)
                {
                    cnt += TraverseSite(n, b.LanguageID);
                }
            }
            //Traverse static files and folders
            var rootPath = HostingEnvironment.MapPath("~/");
            cnt += TraverseFiles(rootPath, "", "*.txt|*.ico", false);
            cnt += TraverseFiles(rootPath, "Static", "*.css|*.js|*.png|*.gif|*.jpg|*.mp4|*.htm|*.html", true);

            //For long running jobs periodically check if stop is signaled and if so stop execution
            if (_stopSignaled)
            {
                return "Stop of job was called";
            }

            return string.Format("Moved {0} items to static web site storage",cnt);
        }
    }
}
```

StaticGeneratorJob.cs hosted with ❤ by GitHub   view raw

Azure   Optimizely (Episerver)

RECENT POSTS

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