



ALLAN THRAEN |

🕒 17 years ago |



A SIMPLE PAGE IMPORT WEB SERVICE



In EPiServer CMS 5 there's a couple of very useful Web Services that gives you pretty thorough access to do just about anything you please - at least with regards to adding / searching / modifying pages.

However the Web Services can be pretty complex and at times a green newbie like myself needs a quick & dirty way of importing files from another system into EPiServer without doing too much thinking. In fact, I found myself in exactly that situation recently, when I wanted to import a huge amount of test data I had scavenged on the net (19040 pages to be exact). So I ended up writing a new little web service that takes the following parameters:

- **ParentID** - The ID of the page that should be the parent of the page you're adding
- **PageType** - The name of the page type to create
- **PageXML** - A string of XML, defining the page. The tag-names should match the page properties you want to set, and they should all be wrapped in a <page> </page> tag. So something like "<page> <pagename>My Page</pagename><bodyField>This is the body</bodyField></page>"
- **Publish** - A boolean parameter that specifies if the page should be published instantly.

The Web method then returns the page-id of the page created, thereby enabling you to build your own page-hierarchy.

In order to make it, I simply created a new standard Web Service in Visual Studio, set the following *Using* clauses:

```
using EPiServer;  
using EPiServer.DataAccess;  
using EPiServer.Core;  
using EPiServer.DataAbstraction;  
using System.Xml;  
using EPiServer.Security;
```

and added this method to the Web Service class:

```
[WebMethod]  
public int ImportPage(int ParentID, string PageType, string PageXML, bool Publish)  
{  
    XmlDocument xd = new XmlDocument();  
    xd.LoadXml(PageXML);  
    PageData pd = DataFactory.Instance.GetDefaultPageData(new PageReference(ParentID),  
    PageType, AccessLevel.NoAccess);  
    //Fill in properties  
    foreach (XmlNode xn in xd.DocumentElement.ChildNodes)  
    {  
        if (xn is XmlElement)  
        {  
            string name = (xn as XmlElement).Name;  
            try  
            {  
                pd[name] = (xn as XmlElement).InnerText;  
            }  
            catch {}  
        }  
    }  
  
    PageReference pr = DataFactory.Instance.Save(pd, (Publish)?  
    SaveAction.Publish : SaveAction.Save, AccessLevel.NoAccess);  
    return pr.ID;  
}
```

So, basically it creates a page with default data, iterates through the 1st level nodes and checks if there is a matching page property. This is a really simple example without any proper error handling, and which only supports string properties. The only thing that's worth noting is how I use the `AccessLevel.NoAccess` in the Save method and the `GetDefaultPageData` method, to avoid uncomfortable access checks (since the web service typically runs as an anonymous user). However, make sure always to put the service behind access-control (which can be set up in web.config).

And yes, the import of my 19k pages went surprisingly well - and pretty quick too!

CodeArt ApS

Teknikerbyen 5, 2830 Virum, Denmark

Email: info@codeart.dk

Phone: +45 26 13 66 96

CVR: 39680688

