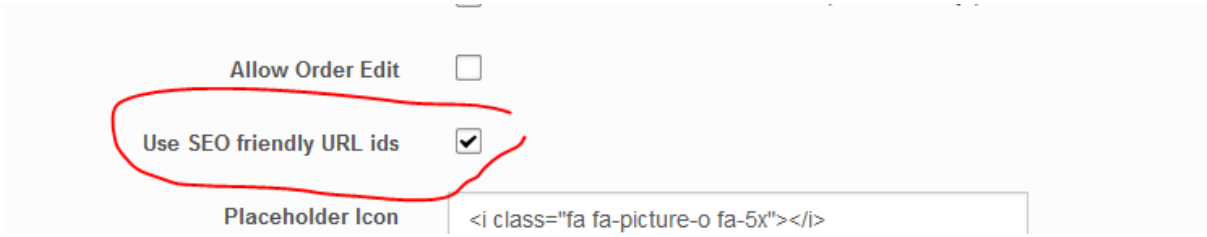
OpenUrlRewriter on NBSv3

Setup/Installation

1. Install NBSv3.5.9 (as per instruction)
2. Install OpenUrlRewriter (https://github.com/sachatrauwaen/OpenUrlRewriter)
3. Install NBrightBuyOpenUrlRewriter
4. IN the NBS Back Office>Admin>Settings select url friendly ids.



Operation

Open Url rewriter has 3 levels of caching activated for NBS.

1. Memory
2. Database
3. File system

The open url rewriter (OURL) module deals with level 1 and 2, so these are not covered in this document. Level 3 is dealt with by the NBrightBuyOpenUrlRewriter (NBSOURL) module.

The file cache for NBS is automatically create when OURL first runs and is stored on the file system under:

. \Portals\X-System\Cache\NBStoreV3UrlRules

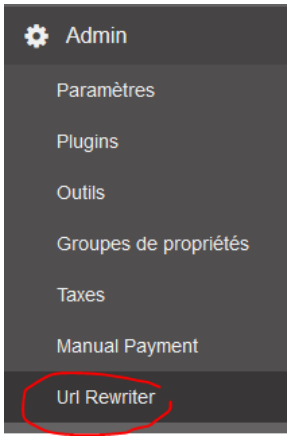
This cache is controlled (removed) by NBS when it updates Products/Categories. And recreated when OURL requests the category/product url, if it has no cache.

The OURL also has DB cache, which it will use before the file cache. Therefore updates in the file cache by NBS will only take effect when the OURL cache is cleared.

THIS MEANS CHANGES OF PRODUCT/CATEGORY URLS DO NOT TAKE EFFECT IMMIDIATELY

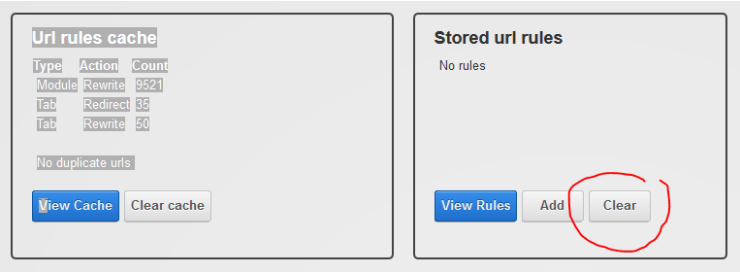
Clearing ALL NBSOURL cache

You can force the NBSOURL to remove ALL file system cache by entering the option in the NBS Back Office. *In normal operation only changed product categories are removed.*

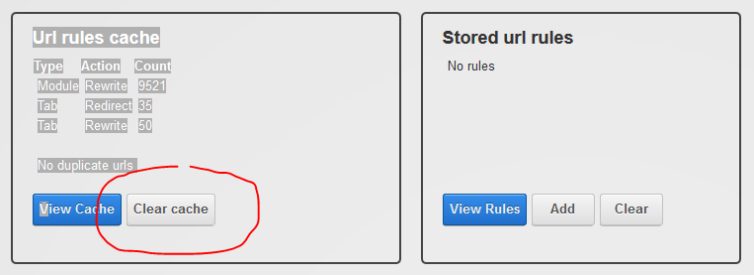


Clearing OURL cache

After clearing the NBSOURL files system cache you can clear the OURL cache and all the new urls will be used. This will be done from time to time periodically or can be done manually by the OURL control interface.



Clears the DB cache.



Clears the Memory cache.

By clearing these 2 cache repositories OURL will use the new NBSOURL filesystem cache changes and repopulate the cache using the latest NBSOURL data.

Code changes made in NBS for OpenUrlRewriter

Changes highlighted: .....

.\Base\NBrightBuyBase.cs

```
protected override void OnInit(EventArgs e)
{

    ModCtrl = new NBrightBuyController();
    DebugMode = StoreSettings.Current.DebugMode;

    base.OnInit(e);

    #region "Get all Settings for module"
    //get Model Level Settings
    ModSettings = new ModSettings(ModuleId, Settings);
    ModuleKey = ModSettings.Get("modref");
    if (String.IsNullOrEmpty(ModuleKey)) ModuleKey = ModSettings.Get("modulekey"); //
keep backward compatiblity with NBS_ProductView.

    #endregion

    if (EnablePaging)
    {
        // SET NOINDEX if we are paging product list
        try
        {
            var page = Utils.RequestQueryStringParam(Context, "page");
            if (page != "")
            {
                var metarobots =
(System.Web.UI.HtmlControls.HtmlMeta)BasePage.Header.FindControl("MetaRobots");
                if (metarobots != null)
                {
                    metarobots.Content = "NOINDEX";
                }
            }
        }
        catch (Exception)
        {
            //ignore
        }

        CtrlPaging = new NBrightCore.controls.PagingCtrl();
        this.Controls.Add(CtrlPaging);
        CtrlPaging.PageChanged += new RepeaterCommandEventHandler(PagingClick);
    }

    //add template provider to NBright Templating

NBrightCore.providers.GenXProviderManager.AddProvider("NBrightBuy,Nevoweb.DNN.NBrightB
uy.render.GenXmlTemplateExt");

    // search for any other NBright Tenmplating providers that might have been added.
    var pluginData = new PluginData(PortalSettings.Current.PortalId);
    var l = pluginData.GetTemplateExtProviders();
```

```
    foreach (var p in l)
    {
        var prov = p.Value;

NBrightCore.providers.GenXProviderManager.AddProvider(prov.GetXmlProperty("genxml/text
box/assembly") + "," + prov.GetXmlProperty("genxml/textbox/namespaceclass"));
    }

}
```

.Componants/NBrightUtils.cs

```
public static string GetEntryUrl(int portalId, string entryid, string modulekey,
string seoname, string tabid, string catid = "", string catref = "")
{

    var rdTabid = -1;
    var objTabInfo = new TabInfo();

    if (Utils.IsNumeric(tabid) && Convert.ToInt32(tabid) > 0) rdTabid =
Convert.ToInt32(tabid);
    if (!Utils.IsNumeric(tabid)) rdTabid = StoreSettings.Current.ProductDetailTabId;
    if ((!Utils.IsNumeric(rdTabid) || rdTabid == -1) && PortalSettings.Current !=
null)
    {
        objTabInfo = PortalSettings.Current.ActiveTab;
        rdTabid = objTabInfo.TabID;
    }

    var cachekey = "entryurl*" + entryid + "*" + catid + "*" + catref + "*" +
modulekey + "*" + rdTabid + "*" + Utils.GetCurrentCulture();
    var urldata = "";
    var chacheData = Utils.GetCache(cachekey);
    if (chacheData != null) return (string)chacheData;


    if (objTabInfo.TabID != rdTabid)
    {
        var portalTabs = NBrightDNN.DnnUtils.GetPortalTabs(portalId);
        if (portalTabs != null)
        {
            var rTab = portalTabs[Convert.ToInt32(rdTabid)];
            if (rTab != null) objTabInfo = rTab;
        }
    }

    // make sure we have the correct culturecode, fo OpenUrlrewriter
    objTabInfo.CultureCode = Utils.GetCurrentCulture();

    var rdModId = "";
    if (modulekey != "") rdModId = "&modkey=" + modulekey;

    var strurl = "~/Default.aspx?tabid=" + rdTabid + rdModId;

    if (StoreSettings.Current.GetBool(StoreSettingKeys.friendlyurlids))
    {
        if (Utils.IsNumeric(catid))
        {
```

```
            var catData = CategoryUtils.GetCategoryData(catid,
Utils.GetCurrentCulture());
            if (!strurl.EndsWith("?")) strurl += "&";
            if (catData.DataLangRecord != null) strurl += "catref=" +
catData.DataLangRecord.GUIDKey;
        }
        if (catref != "")
        {
            if (!strurl.EndsWith("?")) strurl += "&";
            strurl += "catref=" + catref;
        }
        if (Utils.IsNumeric(entryid))
        {
            var prdData = ProductUtils.GetProductData(Convert.ToInt32(entryid),
Utils.GetCurrentCulture());
            if (!strurl.Contains("catref="))
            {
                var defcat = prdData.GetDefaultCategory();
                if (defcat != null && defcat.categoryref != "" &&
!strurl.EndsWith("?"))
                {
                    strurl += "&";
                    strurl += "catref=" + defcat.categoryref;
                }
            }
            if (!strurl.EndsWith("?")) strurl += "&";
            strurl += "ref=" + prdData.DataRecord.GUIDKey;
            seoname = prdData.SEOName;
            seoname = Utils.UrlFriendly(seoname);
        }
    }
    else
    {
        if (Utils.IsNumeric(catid))
        {
            if (!strurl.EndsWith("?")) strurl += "&";
            strurl += "catid=" + catid;
        }

        if (Utils.IsNumeric(entryid))
        {
            if (!strurl.Contains("catid="))
            {
                var prdData = ProductUtils.GetProductData(Convert.ToInt32(entryid),
Utils.GetCurrentCulture());
                var defcat = prdData.GetDefaultCategory();
                if (defcat != null && defcat.categoryid > 0 && !strurl.EndsWith("?"))
                {
                    strurl += "&";
                    strurl += "catid=" + defcat.categoryid;
                }
            }
            if (!strurl.EndsWith("?")) strurl += "&";
            strurl += "eid=" + entryid;
        }
        seoname = Utils.UrlFriendly(seoname);
    }

    urldata =
DotNetNuke.Services.Url.FriendlyUrl.FriendlyUrlProvider.Instance().FriendlyUrl(objTabI
nfo, strurl, seoname);
```

```
    Utils.SetCache(cachekey, urldata);

    return urldata;
}
```

. \Themes\ClassicRazor\Default\NBS_ProductNotFound.cshtml

This File has been added:

```
@inherits NBrightBuy.render.NBrightBuyRazorTokens<NBrightRazor>
@using System.Linq
@using NBrightDNN
@using Nevoweb.DNN.NBrightBuy.Components;

@AddMetaData("resourcepath", "/DesktopModules/NBright/NBrightBuy/App_LocalResources/")

<!-- "pageheaderdetail.cshtml" template will be auto injected as dynamic data into
page head section -->
@{
    // assign Model, so we can resolve var in VS
    var product = (ProductData)Model.List.First();
    var info = (NBrightInfo)product.Info;
}

<div class="addedtobasket"
style="display:none;">@ResourceKey("ProductView.addedtobasket")</div>

<div class="nbs">

    <div class="productdetail" itemscope >

        <h1>Product Not Found</h1>

    </div>

</div>
```

NOTE: This file is using RAZOR, for those system using a version prior to the razor version, you'll need to remove the razor refs and adjust the html to suit your needs.

./ProductRazorView.ascx.cs

```
protected override void Render(HtmlTextWriter writer)
{
    base.Render(writer);
    if (_404code)
    {
        Response.StatusCode = 404;
    }
}

private void RazorDisplayDataEntry(String entryId)
{
    var productData = new ProductData();
    if (Utils.IsNumeric(entryId))
    {
        productData = ProductUtils.GetProductData(Convert.ToInt32(entryId),
Utils.GetCurrentCulture(), true, EntityTypeCode);
    }
```

```
    if (productData.Exists)
    {

        if (PortalSettings.HomeTabId == TabId)
            PageIncludes.IncludeCanonicalLink(Page,
Globals.AddHTTP(PortalSettings.PortalAlias.HTTPAlias)); //home page always default of
site.
        else
            PageIncludes.IncludeCanonicalLink(Page,
NBrightBuyUtils.GetEntryUrl(PortalId, _eid, "", productData.SEOName,
TabId.ToString("")));

        // overwrite SEO data
        if (productData.SEOName != "")
            BasePage.Title = productData.SEOTitle;
        else
            BasePage.Title = productData.ProductName;

        if (productData.SEODescription != "") BasePage.Description =
productData.SEODescription;
        if (productData.SEOTagwords != "") BasePage.KeyWords =
productData.SEOTagwords;

        // if debug , output the xml used.
        if (DebugMode)
productData.Info.XMLDoc.Save(PortalSettings.HomeDirectoryMapPath + "debug_entry.xml");
        // insert page header text
        NBrightBuyUtils.RazorIncludePageHeader(ModuleId, Page,
Path.GetFileNameWithoutExtension(_templD) + "_head" + Path.GetExtension(_templD),
ControlPath, ModSettings.ThemeFolder, ModSettings.Settings(), productData);

        #region "do razor template"

        var strOut = NBrightBuyUtils.RazorTemplRender(_templD, ModuleId,
"productdetailrazor" + ModuleId.ToString() + "*" + entryId, productData, ControlPath,
ModSettings.ThemeFolder, Utils.GetCurrentCulture(), ModSettings.Settings());
        var lit = new Literal();
        lit.Text = strOut;
        phData.Controls.Add(lit);

        #endregion
    }
    else
    {
        _404code = true;
        var strOut = NBrightBuyUtils.RazorTemplRender("NBS_ProductNotFound.cshtml",
ModuleId, "", productData, ControlPath, ModSettings.ThemeFolder,
Utils.GetCurrentCulture(), ModSettings.Settings());
        var lit = new Literal();
        lit.Text = strOut;
        phData.Controls.Add(lit);

    }

}
```

./render/GenXmlTemplateExt.cs

```
private void hrefpagelinkbind(object sender, EventArgs e)
{
```

```
    var l = (Literal)sender;
    var container = (IDataItemContainer)l.NamingContainer;
    try
    {
        l.Visible = visibleStatus.DefaultIfEmpty(true).First();
        var catparam = "";
        var pagename = PortalSettings.Current.ActiveTab.TabName + ".aspx";
        var catid = Utils.RequestParam(HttpContext.Current, "catid");
        if (Utils.IsNumeric(catid))
        {
            pagename = NBrightBuyUtils.GetCurrentPageName(Convert.ToInt32(catid)) +
".aspx";
            catparam = "&catid=" + catid;
        }
        var catref = Utils.RequestParam(HttpContext.Current, "catref");
        if (catref != "")
        {
            catparam = "&catref=" + catref;
            pagename = "";
        }
        var url =
DotNetNuke.Services.Url.FriendlyUrl.FriendlyUrlProvider.Instance().FriendlyUrl(PortalS
ettings.Current.ActiveTab, "~/Default.aspx?tabid=" +
PortalSettings.Current.ActiveTab.TabID.ToString("") + catparam + "&page=" +
Convert.ToString(DataBinder.Eval(container.DataItem, "PageNumber")) + "&pagemid=" +
l.Text, pagename);
        l.Text = "<a href=\"" + url + "\">" +
Convert.ToString(DataBinder.Eval(container.DataItem, "Text")) + "</a>";
    }
    catch (Exception ex)
    {
        l.Text = ex.ToString();
    }
}
```