Using Apollo PageLocalization to localize your menu

A brief guide

Revision history
May 23, 2005 – for Apollo.LocalizationApi 1.1
June 12, 2005 – for Apollo.LocalizationApi 2.0
October 22, 2008 – for Apollo.LocalizationApi 3.0

Adding localization to your menu with PageLocalization is pretty simple, it involves not much more than adding a reference to Apollo.LocalizationApi to your project (when using early binding/strict on), and adding (or rather: changing) one line of code of your menu rendering method. When you localize your tabs, make sure you do this at the earliest available moment in your code, as not only the name and title is localized, but also whether the tab is visible for the selected locale (with the property IsVisible). So your tab selection routine should happen after localization.

You are allowed to add the Apollo.LocalizationApi.dll to your project, and distribute it with your private assembly, or with your source code. Alternatively you can use late binding, in which case the api dll does not have to be distributed with your project. The localization api will check whether PageLocalization is available. If PageLocalization is not available, then you will keep working with an unlocalized tabinfo object.
The localization api determines the availability of PageLocalization based on the existence of 2 key objects in the database, the Apollo_TabLocalization table, and a stored procedure.
You are allowed to distribute the *sourcecode* of the api with ***source*** versions of your api. You must include the complete api project, with helpfile, extra license text and this tutorial.

Creating a localized Links skinobject – using early binding (option strict on)

The following example assumes that you have at least some prior experience developing modules and/or skinobjects for DotNetNuke, making use of Visual Studio. For this example I am using DNN 4.9.0 and Visual Studio 2008 and the DNN Starterkit.

Let me start by creating a new skinobject, using the WSP (website project) approach. I will call it TestMLMenu.

- Open your DotNetNuke website from within Visual Studio
- Create a new folder named "TestMLMenu" in the DesktopModules folder
- Copy the files Links.ascx and Links.ascx.vb from the folder /Admin/Skins/ to the newly created folder /DesktopModules/TestMLMenu/
- Rename the file Links.ascx TestMLMenu.ascx
- Open the file TestMLMenu.ascx in source view, and replace
  ```
  Inherits="DotNetNuke.UI.Skins.Controls.Links"
  ```
  with
  ```
  Inherits="YourCompany.SkinObjects.TestMLMenu"
  ```
  and save the file
- Open the file TestMLMenu.ascx.vb, and replace
  ```
  Namespace DotNetNuke.UI.Skins.Controls
  ```
  with
  ```
  Namespace YourCompany.SkinObjects
  ```
  and replace
  ```
  Partial Class Links
  ```
  with
  ```
  Partial Class TestMLMenu
  ```

- Add “Imports DotNetNuke” at the imports section of the file
- Add “Imports Apollo.DNN_Localization” at the imports section of the file

Now we add the localization code. Because the LocalizedTab class publishes the same properties as the core TabInfo class (in fact the class inherits from TabInfo), we don’t need to do much. Somewhere around line 151 (if you followed the exact same procedure), you will find this line of code:

```
      sbLinks.Append(ProcessLink(ProcessTab(CType(PortalSettings.Deskto
      pTabs(intIndex), TabInfo), strLevel, strCssClass),
      sbLinks.ToString.Length))
```

Replace this line of code with:

```
      Dim tab As Apollo.DNN_Localization.LocalizedTabInfo =
Apollo.DNN_Localization.LocalizeTab.getLocalizedTab(CType(PortalSettings.Deskto
pTabs(i), TabInfo))
```

This is all! This line of code gets the needed tab, just like in the orginal version, and passes this to the shared function getLocalizedTab. This function will always return something (unless the passed tab was also Nothing). If no localized version of the tab is available, a dummy localized tab will be made, and all it’s values will be initialized with the values of the original tab.

In order to use the skinobject, you must register it in DNN. Go to Host>ModuleDefinitions, and select the [SKINOBJECTS] item. Scroll to the bottom of the page, and click “Add Control”. In the following form, enter “TestMLMenu” for key and select /DesktopModules/TestMLMenu/TestMLMenu.ascx for source. Click “Update”.
Next, to use the skinobject in a skin, do this (using the ascx version of the skin):

- Add the following registration to the top of the page
  <%@ Register TagPrefix="yourcompany " TagName="TestMLMenu" Src="~/DesktopModules/TestMLMenu/TestMLMenu.ascx" %>
- Add the following control declaration somewhere in your skin:
  <yourcompany:TestMLMenu runat="server" ID="MLLINKS" CssClass="links" Level="Root" Separator="&nbsp;&nbsp;&nbsp;|&nbsp;&nbsp;&nbsp;" />

If you want to create a PA installer zip file for this, add the files TestMLMenu.ascx and TestMLMenu.ascx.vb to a zipfile, and include a TestMLMenu.DNN file with the following contents:

```
<?xml version="1.0" encoding="utf-8" ?>
<dotnetnuke version="2.0" type="SkinObject">
    <folders>
        <folder>
            <name>TestMLMenu</name>
            <modules>
                <module>
                    <controls>
                        <control>
                            <key>TestMLMenu</key>
                            <src>TestMLMenu.ascx</src>
                            <type>SkinObject</type>
                        </control>
                    </controls>
                </module>
            </modules>
            <files>
                <file>
                    <name>TestMLMenu.ascx</name>
                </file>
            </files>
        </folder>
    </folders>
</dotnetnuke>
```

This will make the zipfile into a SkinObject package which can be installed in any DNN installation

Creating a localizable menu – using late binding (option strict off)

When using late binding of Option Strict Off, there is no requirement to add a reference to the LocalizationApi.dll to your project. However, localizing your tabs will be a bit more work, as you will have to check yourself whether the api is available. You can do this by adding the following function to your class:

```
        Private Function LocalizeTab(ByVal objTab As
DotNetNuke.Entities.Tabs.TabInfo) As DotNetNuke.Entities.Tabs.TabInfo
        Try
                Dim objLocalization As Object
                objLocalization =
        System.Activator.CreateInstance("Apollo.LocalizationApi",
        "Apollo.DNN_Localization.LocalizeTab").Unwrap ' New
        Apollo.DNN_Localization.LocalizeTab
                Return objLocalization.getLocalizedTab(objTab)
        Catch exc As Exception
                Return objTab
        End Try
        End Function
```

This function does the following:

- First a new instance of Apollo.DNN_Localization.LocalizeTab is created, making use of system.activator, based on the assemblyname and the classname.
- If this call fails (when the api is not available), the fallback method (in the catch statement) gets executed.
- If the call succeeds, a localized tab is returned. Mind you, this only works with late binding!

Now the localization of a tab is easy, just make this call:

```
        Dim Tab As TabInfo = LocalizeTab(CType(PortalSettings.DesktopTabs(i),
TabInfo))
```