N2 CMS

3/31/2010

The Little Handbook

This document will be an ongoing project for me, as I discover about N2CMS. It's a little handbook that I am writing as a help for myself, as I try to understand the magic behind N2.
I am a seasoned professional software developer but I am mostly a newbie in regards to N2CMS, so keep in mind that I may very well make **mistakes** in this document. In fact, I will most certainly do! So stay alert, don't get mad at me ;) and don't complain to anyone else: all errors are **mine**.
In any case, this is my first contribution to any kind of open source project and I hope that it helps you. I also want to express my thanks to N2's development team for providing us with a totally **outstanding** piece of software for free (as far as I know, this means *Cristian Libardo* and Steve Mason.)

Jacques Ronaldi
Software Developer
jacques@interferencelogik.com

Contents

ASP.NET

N2 CMS

THE LITTLE HANDBOOK

INTRODUCTION

After installing the N2CMS software system, I quickly realized that I had gotten two things:

- A great system to build custom content manageable websites;
- More than I bargained for…

Indeed, N2CMS was obviously very powerful, but also lacking much in documentation. If I was to use N2CMS, it wouldn't come for free: I would have to work for it. So after thinking about it and revisiting all of the alternative software, I finally made a decision that : it was worth my time.

Indeed, it quickly became **obvious** that "these guys" were serious software developers. The software seemed huge but well crafted and I was certain to **learn** quite a few things while trying to make sense of N2CMS.

The final blow, the "thing" that really convinced me that N2 was the right choice was when I saw "Swedish" references here and there… What? This software is Swedish?? No way! I was backpacking in Sweden for a whole month right that summer of 2009 and I simply fell in **love** with Stockholm (jag är också lära sig svenska!) I thought this had to be a "sign from destiny", ha ha ha. Whatever… I just **had** to learn everything I could about this Swedish software :)

In any case, where there's a fool, there's probably two or more… so I'm probably not alone trying to make sense of this masterful piece of work. Why not share my discoveries? And so started this small handbook that I hope will help everyone as much as myself.

Lycka till!

SITE STRUCTURE

Overriding Templates

Very quickly, it seems many people try to play with the system's page and master page structure for many reasons. I did the same thing and just like it happened for many, I had many compile errors and weird runtime behaviors until I started to understand "some basics".

> **When you create new files, try not to create any pages with the duplicate names, even if they are in separate folders. It seems N2's URL rewriter does not remap a URL if a file exists at the provided location. So this can lead to weird behaviors where the page being used is not the one that you think.**

Creating Entirely Separate Start Page, Page and Master Page

Let's start with the "ultimate" disconnection from N2's automated system: the creation of a custom "start page" and a custom "default page" both using a custom master page (the same for both.)

In that case, what you do is create master page "*MasterX*", with the usual "content panes" inside it. The code part of this master page will inherit from the usual "*MasterPage*".

Then you create a "start page" as "start.aspx", for example, and set its master page name to reference the above "*MasterX*" and so will probably reference the master page's content panes. The code part will need to derive from a N2.Web.UI.ContentPage<*StartPageItem*>. You must not use N2.Templates.Web.UI.TemplatePage<*StartPageItem*> because this would cause the master page file designation to be overridden by the system. We rather want to have our custom page use our specific custom master page.

You will need to implement *StartPageItem* in the App_Code folder (do not be tempted to just add this quick class right after your custom "Page" implementation, it wouldn't be found by N2's install system!)

The *StartPageItem* will need some special attribute markers in order to advise N2 that this is a new "start page" item. It also will derive from *LanguageRoot* as this will be required for example, to have a Google UrChin tracker part on the start page (that part validates that the parent is a *LanguageRoot*.

```
code
```

After this, you create the new custom "default page" in file *default.aspx*. This will have the same code as

If you create a new custom page that references a custom master page, then the "code" of the custom page must not derive from TemplatePage<>. This would make it use the one configured. Rather, derive ***

Making a website multilingual

Making a website multilingual with N2 CMS is actually very simple – once you understand how to. Indeed, I was baffled for a while, having "half working" solutions that would "mostly work". But I would get weird behaviours, such as having 1 french flag and 2 english flags show up at the top of my website. Why 2 english flags? And I wasn't alone with the issue, I've found live websites on the Internet showing this problem so I knew I was not alone. Should that be comforting me?

As far as I can tell, the problem with having multiple language flags appearing on your website is caused by having more than one item deriving from "LanguageRoot" for that language. There should be a single *LanguageRoot* per language. Seems easy enough, but when starting off with N2 and trying to build custom templates and stuff, it's very easy to miss that something ultimately derives from a *LanguageRoot*.

Also, I did the mistake of having the language root at the… *root* (!) of my website, I mean, at the same level as my "*start page*". That's wrong. In spite of the word "root" being part of "*LanguageRoot*", it refers to the "root" of a new set of pages for a specific language, not the "root" of the website!

So the "*LanguageRoot*" in fact goes under the "start page", which itself goes under the website's "root" (a page that is never really served to the world.)

```
Root Page
+-----  Start page (deriving from a LanguageRoot an set to a specific language)
        +-----  Page 1 (in the same language as "Start page")
        +-----  Start page (deriving from a LanguageRoot and set to a different language)
                +-----  Page 1 (translated from page 1 above, using N2's UI interface)
```

Each of the pages in the above example has its own "URL segment" that will determine the actual path to any page. Assuming each of the above pages would use the same "URL segment" as their page name, then reaching "Page 1" for the 2nd language of that website, for example "French", would be accomplished with the following URL:

/Start-page/Start-page/Page-1

COMMON ERRORS

Deployment

Error 404 – page not found

This error can occur when deploying on an IIS server if the extension-less "*" page request URLs are not mapped to the ASP.NET handler. Either map these in the mapping entries, or else the pages will have to be rewritten with a "/segment/…/default.aspx" kind of URL (with ".aspx" making the mapping work for IIS.)

Another possibility for the error is if the website is running in a legacy "compatibility application pool". Configure the website to use the default IIS7 pipeline mode and everything will start working.

Language flags not appearing or cannot enter "/edit" website administrative pages

As part of deploying the website, the folders "/edit/*" and "/Templates/*" folders must be copied to the server.

Error 401 – unauthorized

If the "unprotected" files are inaccessible, check in authentication provider for "anonymous authentication", "edit" and make sure the credentials are for a user that has access. It is recommended to set "use application pool credentials" and ensure that the application pool has credentials set for an authorized user, such as "NETWORK SERVICE".

N2CMS MVC

INTRODUCTION

This section pertains to information specific to the MVC version of N2. Also, this was written using an early **alpha version** of N2 MVC, so things might have changed (N2 MVC currently has a Beta version recently made available.)

INSTALLING N2 CMS 2.0 BETA 2

Here's a startup guide to installing the latest beta (as of this writing) of N2 CMS 2.0.

Getting the N2 source file

The link http://n2cms.codeplex.com/releases/ brings you to the download section, then selecting the latest version from the "releases" box on the right should get you to the most current version. To use the same beta as the one used for this guide, download 2.0 Beta 2 at http://n2cms.codeplex.com/releases/view/42703 called "N2 CMS 2.0 Beta2 Sources".

What you get after download is a zip file, "N2_CMS_2.0_beta2_Sources.zip". Unzip it somewhere, getting folder "N2 CMS 2.0 beta2 Sources" in the process.

Preparing and compiling N2

First thing to do is execute "Prepare_Dependencies-vs2008.bat" in that folder. You can open a CMD box to that folder and execute the command file from there, or simply go there with Windows Explorer and double-click on the file, which will do the same thing.

This process takes less than a minute and produces a lot of colored lines flying by on the screen as the various logs and statuses are displayed by the preparation process. Unless you see some **red**, you should be fine :)

Next step is to actually compile our brand new N2 source code! For that, simply double-click on the solution file, "N2.Everything-vs2008.sln". I mean, you already **have** Visual Studio 2008 installed, right?

And also, did you install *Visual Studio SP1*? Search ""VStudio sp1" on Google, or get it at http://www.microsoft.com/downloads/en/confirmation.aspx?familyId=fbee1648-7106-44a7-9649-6d9f6d58056e&displayLang=en. You need it for MVC 2.

Next? Well, if you follow closely you just read that we need MVC 2... so Google "MVC 2 RTM" and download... (the link is just too long to paste here!)

Okay now. All these software have been installed, you should be ready to go. If you get an error when you open "N2.Everything-vs2008.sln" complaining about an unknown project type, then it means you did **not** install all of the above software (mainly, you're missing MVC 2, so follow the steps above).

Right-click on the solution file in solution explorer and "build". All should be compiling nicely and if so, now you are ready to try running the great sample website that comes with N2 MVC. Find and right-click on the project "N2.Templates.Mvc" under MVC folder and select "Set as startup project".

Select "Debug -> Start without debugging" in the main menu or press Ctrl+F5 to run the website sample. Does it pop up? It should... otherwise stop and review the above steps.

ROUTING

N2 will load the page to be rendered by looking up the correct page at the specified URL from the database and then use the normal routing tables to determine the proper controller to use.

This means that a matching controller for a request is found according to the routes that have been setup. The matching process between URL and routes either finds a specific matching controller, or uses the default controller if none is found.

An interesting twist with N2 is that a specific controller can override the default URL/route matching and force a match for a specific content page type! A controller forces a match by adding a "Controls" attribute for any number of 'content page types' for which the controller applies.

```
[N2.Web.Controls(typeof(SecondaryPage))]
public class SecondaryController : ContentController<SecondaryPage>
```

In the above case, the "SecondaryController" will be selected when a page of type "SecondaryPage" is invoked, regardless of the URL.

Here's another example where a controller will be used for either a 'content page type' of "SecondaryPage" or "AnotherTypeOfPage". Note the use of 'N2.ContentItem' as a generic type in this case, since we don't know which kind of page item will be matched (because we support many.)

```
[N2.Web.Controls(typeof(SecondaryPage))]
[N2.Web.Controls(typeof(AnotherTypeOfPage))]
public class SecondaryController : ContentController<N2.ContentItem>
```

N2 Page Item used by controller actions

An N2 page **must** exist in the database at the received URL for N2 to properly map the controller's current page item. That means a specific "page type" (model) must be declared for each controller and used to create a page in the database that has the proper controller's URL (without MVC "actions".)

The controller then has to specify that page item as the type of it's [Controls(typeof(custom-page-type))]. This allows N2 to match the correct controller for a page type and load the correct "CurrentPage" item. This is critical if the views use an N2 master page, for example. It would not be useful for a page view that doesn't reference N2's *CurrentPage* or *CurrentItem*.

Remember that the **view** that will be loaded for the content page type item will be determined by the selected controller. If no custom controller matches the content page type, then N2 will provide a default controller. The path to the content page item's view can then be overridden by adding the *MvcConventionTemplate* attribute to a specific content page type, as followed.

```
[N2.Web.Mvc.MvcConventionTemplate("SomePath")]
public class AcctAdminPage : AbstractContentPage, IStructuralPage
{}
```

In the above example, N2's controller would try looking for the view to load into the "~/Views/SomePath/" location instead of "~/Views/AcctAdminPage/" which would be the default in the absence of a custom controller.

Note:

A specific page is not required to define its own custom page type if the base controller's page item fits the requirements. As long as a default view ("Index.aspx") exists for the matching controller, N2 will properly load that view for the controller's page item.

The supplied controller action will thus be activated with the default controller action's page item. This allows controller action views to refer to an N2-enabled master page without having to define useless page items for each and every action.

N2 Page Item view path

The **view** that is loaded for the content page type item will be determined by the selected controller. If no custom controller matches the content page type, then N2 will provide a default controller. N2 will automatically look for a matching view of the same name as the content page type in the standard locations.

The path to the content page item's view can also be overridden by adding the *MvcConventionTemplate* attribute to a specific content page type, as followed.

```
[N2.Web.Mvc.MvcConventionTemplate("SomePath")]
public class AcctAdminPage : AbstractContentPage, IStructuralPage
{}
```

In the above example, N2's controller would try looking for the view to load into the "~/Views/SomePath/" location instead of "~/Views/AcctAdminPage/" which would be the default in the absence of a custom controller. This attributes on works with N2's controller. The attribute does not affect the path that would be selected if a custom controller is found (and used intead of N2's) for the content page type.

COMMON ERRORS

Page Exceptions

Object reference not set to an instance of an object.
caused by null object reference accessing *ContentItem* or *ContentPage* with N2 MVC

This error can occur for many reasons, one of which is attempting to render a page that has a proper controller, as well as a proper view, but lacks an equivalent page in N2's database that matches the

controller's path. For example, if you have a controller at "/controller/action", then you would need a root page with URL "/controller".