# Localization Guide for DotNetNuke Modules

## Testing module localizability

Vicenç Masanas


Version 1.0.0
Last Updated: March 20, 2006
Category: DotNetNuke v3.x, v4.x

*Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this document remains with the user.*

*The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred.*

*Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Perpetual Motion Interactive Systems, Inc. Perpetual Motion Interactive Systems may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Perpetual Motion, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.*



*DotNetNuke® and the DotNetNuke logo are either registered trademarks or trademarks of Perpetual Motion Interactive Systems, Inc. in the United States and/or other countries.*

*The names of actual companies and products mentioned herein may be the trademarks of their respective owners.*

# Abstract

**This document contains a list of items to verify to assure a module supports as many DotNetNuke localization features as available on the Core Framework.**

# Contents

# Why localize?

## Introduction

Since version 3 DotNetNuke provides a rich set of features to allow static localizability of modules. By static information we mean all data that is not subject to user modification and that can be considered for the most part as not changeable. This includes text labels, error and information messages, help, and graphics that are spread throughout the user interface of the application.

As DNN is used in more and more countries it is very important to provide modules with good support for international users. Otherwise we are limiting our potential user base to that using English as their base language.

Through the rest of this document you will get to know some of the most common problems found in many modules and how to solve them.

# Database

## Data types

Always use data types that support Unicode data. In SQL Server these are:

- nchar

- nvarchar

- ntext

Verify all character fields use these data types instead of *char, varchar* and *text*.

### Where to check

- Database scripts included in the module install package. The source for these files is usually found in

- \<*dnn installation folder*>\DesktopModules\<*Module Name*>\Providers\DataProviders\SqlDataProvider*.SqlDataProvider

### What to check

- Fields defined in CREATE TABLE statements

- STORED PROCEDURE parameters

- Variables defined in stored procedures (DECLARE statements)

# Collation clauses

If you use SQL Server default features to write the scripts that will create the module tables you will usually see that it includes collation clauses for every database field.

Unless you want to force the collation on any given field to something specific it is good practice to remove all collation clauses in the table creation scripts in order to force it to use the default collation used by the database. This way, the module setup will not overwrite database defaults and will use whatever setup the database administrator has decided.

## Where to check

- Database scripts included in the module install package. The source for these files is usually found in

\<*dnn installation folder*>\DesktopModules\<*Module Name*>\
Providers\DataProviders\SqlDataProvider*.SqlDataProvider

## What to check

- COLLATE clauses in table creation scripts.

A sample bad code is:

```
[Title] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
```

This should be corrected to:

```
[Title] [nvarchar] (255) NOT NULL
```

# DotNetNuke Controls

## Using built-in localization features

Normal DNN controls will inherit from **PortalModuleBase,** which has built-in support for localization. If the module contains any extra user controls they should inherit from **UserControlBase** to be able to use the same localization features. There is also a **PageBase** class that can be used to provide localization to any extra aspx page the module contains.

### Where to check

- All user controls and pages (*.ascx, *.aspx)

### What to check

- Classes inherit from DNN base classes (PortalModuleBase, UserControlBase, and PageBase)

## Display missing keys

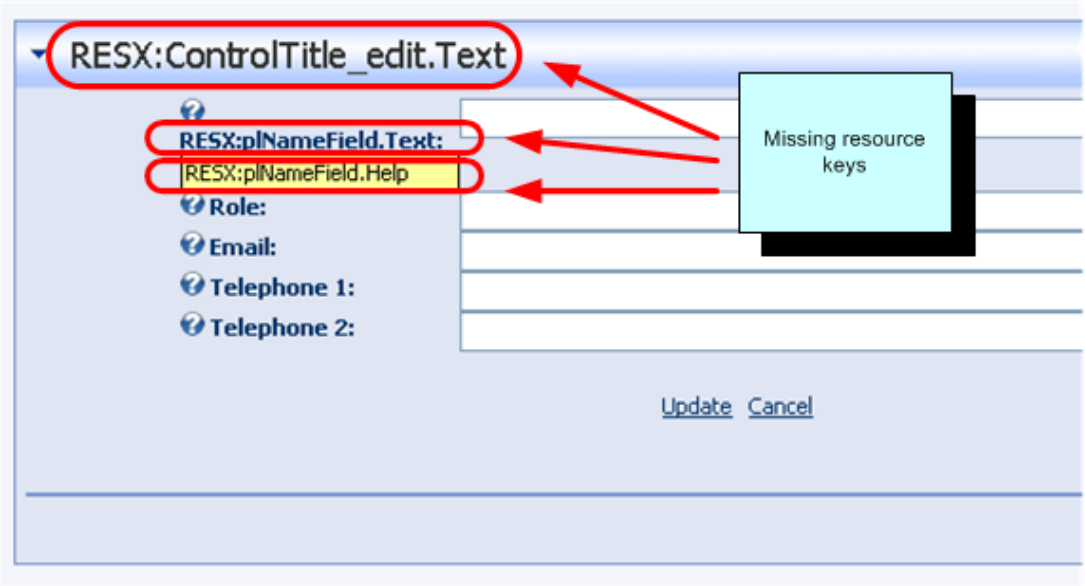
Each DotNetNuke control in the module should include its own resource file for localizable items. If there are many items in these files it is usually difficult to keep them synchronized with the code.

To be able to detect any missing resource keys during development DotNetNuke provides a way to clearly identify missing resources in any file. Open web.config and set the ShowMissingKeys key to true:

```
    <add key="ShowMissingKeys" value="true" />
    <!-- Show missing translation keys (for development use) -->
```

This will display a message at run time for any missing resourcekey found in any control.

This flag should only be used in development or testing installations. It is recommended that you set it to false on the production installation.




### Where to check

- web.config

### What to check

- ShowMissingKeys value

## Localize text labels

DNN includes a useful control to easily integrate both help and localization for text labels. These are usually found next to an input control where users have to enter some text or interact with.

To use this control you need to add a reference to the control:

```
<%@ Register TagPrefix="dnn" TagName="Label" Src="~/controls/LabelControl.ascx" %>
```

and then use it as with any other user control.

```
<dnn:label id="Title" runat="server" controlname="txtTitle"></dnn:label></td>
```

When using this user control the *id* will be used as a key to find the appropriate resource string in the resource file:

- <key>.Text: the text that will be displayed

- <key>.Help: the help text that will be associated with this label

Following the previous example, the resource keys associated with the label will be:

```
    <data name="Title.Text">
        <value>Title</value>
    </data>
    <data name="Title.Help">
        <value>Enter a Title for ...</value>
    </data>
```

> Note that resource keys are case sensitive.

> A resource key will be searched on the resource file with a **".Text"** appended to it by default if no other *extension* is used. ".Text" is the default extension for resources and can be omitted when referenced in design files or in code.

You can localize other texts on most ASP.NET server controls by adding a "resourcekey=" attribute to the control and providing the key translation in the resource file.

```
<asp:hyperlink id="cmdCalendar" resourcekey="Calendar" Runat="server"> </asp:hyperlink>
```

When checking if the module includes resource keys for all text labels the best approach is to translate the module to a different language and make it the default language for the active user.

If you see any message in English it means you forgot to localize some text.

If you see any message similar to **RESX: Title.Text** it means you included a resource key in some control but the key is not present in the associated resource file.

### Where to check

- All user controls (*.ascx files)

- All external pages (*.aspx files)

- Resource files: found in \<*dnn installation folder*>\DesktopModules\<*Module Name*>\AppLocalResources

### What to check

- Label, Checkbox, RadioButton, Button, HyperLink, LinkButton, ... controls missing a *resourcekey* attribute

- Images missing a *resourcekey* attribute if they provide an “Alt” text

- RadioButtonList, CheckBoxList, ListBox and DropDownListBox controls include a *resourcekey* for all option items statically added

- Validator controls missing a *resourcekey* for the error message(s) to be displayed

- Use of DNN LabelControl where appropriate

## Module Control Title

Each module control that is accessed in “edit” mode displays a fixed Module Title initially configured in the Module Definition section. Since this title is fixed it can also be localized using resource files.

You don't need to add any extra code to support this feature, just the appropriate keys in the correct resource files.

If a module has a control "modulecontrol.ascx" accessed through a control key "*akey*" all you need to do is add the following resource to the modulecontrol.ascx.resx resource file with the desired text for the title and the DNN framework will read the title when opening this control.

```
ControlTitle_akey.Text
```

> Note that the control key in the resource must be lowercase.

### Where to check

- All user controls (*.ascx files) that will be opened in "edit" mode

- Resource files: found in \<*dnn installation folder*>\DesktopModules\<*Module Name*>\AppLocalResources

### What to check

- Resource files contain a resource key for the control key used to access the control in the form of "ControlTitle_*controlkey*.Text"

## Datagrids

Controls in datagrid templates can be localized using the same approach as discussed in other sections of this document but the column headings need a special process in order to be localized. DotNetNuke.Services.Localization.Localization class includes a method **LocalizeDataGrid** that can be used to localize datagrid column headers.

To localize column headers you need to include a resource key for each column in the datagrid with the same key as the *HeaderText* attribute of the column followed by ".Header".

For example:

```
<asp:datagrid id="dgSample" runat="server" >
<Columns>
<asp:TemplateColumn HeaderText="Title">
...
</asp:TemplateColumn>
</asp:datagrid>
```

can be localized by including a resource key “Title.Header” with the desired text for the heading and calling this method:

```
Localization.LocalizeDataGrid(dgSample, LocalResourceFile)
```

You have to call the method before binding the data to the datagrid.

### Where to check

- All controls that include a DataGrid

### What to check

- Related resource file includes resource keys for all column headers

- LocalizeDataGrid is called in code behind file

## JavaScript Messages

String messages used in JavaScript code need to be escaped in order to preserve correct interpretation of JavaScript special characters. For example [ ‘ ] (quote) in JavaScript is used as a string delimiter, so if you want to add a message containing a quote you have to escape it: \’

There are two methods in the class **DotNetNuke.UI.Utilities.ClientAPI** that can help in the process of providing JavaScript with the correctly formatted strings.

- AddButtonConfirm: can be used to handle confirmation prompts at the client
- AddSafeJSString: escapes strings to be used in client JavaScript

To use these methods you have to include a reference to DotNetNuke.WebUtility in your project.

Sample code:

```
ClientAPI.AddButtonConfirm(lnkDelete, Localization.GetString("Delete",LocalResourceFile))
```

```
<script language="javascript">
//Localization Vars
var sOk =
'<%=ClientAPI.GetSafeJSString(Localization.GetString("Ok",LocalResourceFile))%>';
...
```

### Where to check

- All files where JavaScript is used to display or change any text at the client

### What to check

- Confirmation button messages

- Dynamically created JavaScript code

## Messages on code

On a number of occasions a module needs to display a message, or send an email, or change a text, ... based on certain conditions. These texts cannot be localized using the resource key approach used in the design files (*.ascx, *.aspx).

DotNetNuke provides a rich API for accessing localized strings from resource files using a basic set of methods.

Each message should be included in the appropriate resource file and associated with a resource key. The resource key will be used to get the actual message from the resource file at runtime based on the active language for the user.

Sample bad code:

```
lblTitle.Text = "The title"
Dim strMessage as String = "A message"
Dim strResponse as String = "Hello " + UserInfo.FullName + ", and welcome to our site"
```

Which should be corrected to:

```
Imports DotNetNuke.Services.Localization
lblTitle.Text = Localization.GetString("Title", LocalResourceFile)
Dim strMessage as String = Localization.GetString("Message", LocalResourceFile)
Dim strResponse as String = String.Format(Localization.GetString("Response",
LocalResourceFile),UserInfo.FullName)
```

And the related resources added to the resource file:

| Resource Key | Value |
|---|---|
| Title.Text | The title |
| Message.Text | A message |
| Response.Text | Hello {0}, and welcome to our site |

If you need to get a message from a class that is not directly tied to a visual control (user control or page), for example a component class, you have to include the resources for this messages in a special resource file (*SharedRecources.resx*) and specify this file in the method call.

You can also use this feature to centralize in a single file common messages that are used in many places in the module, preventing to have to localize the same messages in more than one file.

Sample bad code:

```
Class ObjectController
  Public Function AMethod() As String
   return "a message"
  End Function
End Class
```

Which should be corrected to:

```
Class ObjectController
  Private SharedResourceFile As String = ApplicationPath +
"/DesktopModules/<ModuleFolder>/App_LocalResources/SharedResources.resx"

  Public Function AMethod() As String
   return Localization.GetString("Message", SharedResourceFile)
  End Function
End Class
```

And the related resources added to the resource file **SharedResources.resx** :

| Resource Key | Value |
|---|---|
| Message.Text | a message |

## Where to check

- Code behind files

- Component classes

## What to check

- All message strings are exported to a resource file and accessed through Localization class methods.

# DotNetNuke Global Shared Resources

Before starting to add resource keys and texts to your module's resource files take a look at the file **\App_GlobalResource\SharedResources.resx**. This file contains a number of common messages used in most user interactions that are centralized in a single place to reduce the localization effort. By using these resources you can be sure your module will use a common interface since these resources will be provided by the default language pack installed. And you will also save people localizing your module from having to localize these texts again.

Common items to reuse from SharedResources file include:

- Texts for buttons: cmdBack, cmdCancel, cmdDelete, cmdDisplay, cmdUpdate, ...
- Date, time expressions: Hour, Minute, Day, ...
- Navigation: Last, Next, Previous, First, ...

## Where to check

- \App_GlobalResource\SharedResources.resx

- Module resource files

## What to check

- Duplicated entries that can be removed from module resource files

# Module Help

DotNetNuke includes built-in support for displaying a help screen associated with each control. If you use this feature in your modules, your help will automatically be localizable since this feature is based on resource files. You can use this feature to add detailed user instructions on each control contained by your module.

To provide localizable help for you module just add a resource entry with the key "**Module.Help**" to the resource file associated with each control.

### Where to check

- DotNetNuke control resource files

### What to check

- Resource files include a Module.Help key with detailed help for the control

## Action Menu Items

Navigation amongst module controls is usually based on Action Menu items (commonly displayed as a menu in the top-left corner of the Module title).

Titles for action menus should also be localized using default DNN methods.

Sample code:

```
Public ReadOnly Property ModuleActions() As ModuleActionCollection Implements
      Entities.Modules.IActionable.ModuleActions
   Get
      Dim Actions As New ModuleActionCollection
      Actions.Add(GetNextActionID, Localization.GetString("Add.Text", LocalResourceFile),
...)
      Return Actions
   End Get
End Property
```

### Where to check

- Code behind files for module controls that have action menus

### What to check

- Action titles are exported to resource files

# Images

Images should not contain text because the DNN localization framework does not currently support handling of different sets of images per locale.

If you include images with text in your modules you are forcing users to create an anternate set of images to use in their language, thus making more difficult to translate the module.

If you want to include images with text you should provide a way to change the images dinamically depending on the active language.

A possible approach is providing a resource key for the image file so the user can customize the image filename for each different language:

```
TheImage.ImageUrl = "<custom path for images>\" &
   Localization.GetString("ImageFile",LocalResourceFile)
```

## Where to check

- Image files

## What to check

- Images with text should be replaced so they don't include any specific language string

# Additional Information

- For more information on DotNetNuke localization capabilities read the document titled “DotNetNuke Localization” included in the base documentation package available in the downloads section on www.dotnetnuke.com.

- If you need more details on the localization features of DotNetNuke and how the specific methods are implemented you can look at the class **DotNetNuke.Services.Localization.Localization**

## Errors and Omissions

If you discover any errors or omissions in this document, please email marketing@dotnetnuke.com. Please provide the title of the document, the page number of the error and the corrected content along with any additional information that will help us in correcting the error.

# Appendix A: Document History

| Version | Last Update | Author(s) | Changes |
|---|---|---|---|
| 1.0.0 | Mar. 20, 2006 | Vicenç Masanas | • Initial Version |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |