DotNetNuke Client API MinMax

Jon Henning

Version 1.0.0
Last Updated: June 20, 2006
Category: Client API

DotNetNuke Client API MinMax

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

In order to clarify the intellectual property license granted with contributions of software from any person or entity (the "Contributor"), Perpetual Motion Interactive Systems Inc. must have a Contributor License Agreement on file that has been signed by the Contributor.

Contents

DotNetNuke Client API-MaxMin

Introduction

Utilizing the DotNetNuke Client API enables the developer to offer a rich client-side feature-set. This document outlines one of those feature-sets; providing the ability to maximize and minimize modules without using postbacks.

Goals

Communicate and document the steps involved in creating rich client-side feature-set.

Provide a better understanding of the inner-workings of the Max/Min feature-set.

EnableMinMax Wrapper Function

My original focus in creating the ClientAPI was to give the module developer a framework to free them from having to learn the different techniques to effectively write cross-browser code. Instead they could focus on writing the functionality they deemed appropriate. In order to demonstrate using the ClientAPI I provided a few enhancements like MinMax and Drag-n-Drop. The documentation of these enhancements was to give the module developer some insights into the different ways the API could be used.

After receiving many emails asking how does the ClientAPI allow me to easily do something like MinMax I realized that while my goal of giving examples was accomplished, but I fell short in the area of easily allowing my enhancements to be re-used. This is something that I plan on addressing in upcoming releases, starting with the introduction of the `EnableMinMax` function.

Currently, the DotNetNuke code contains three different ways to handle the MinMax functionality. The visibility.ascx user control used in the modules utilizes cookies to persist the user's preference. The SectionHeadControl.ascx control utilizes the ClientAPI variables to persist the MinMax state between page posts, and finally the

HelpButtonControl.ascx doesn't persist any state, thus with each page refresh the help returns to its minimized view. Additionally, the visibility and section controls utilize images (+/-) to show the action that will be taken on the control when clicked, whereas the help always shows the help icon. All of these factors are now handled in the `EnableMinMax` method.

```
Public Shared Sub EnableMinMax(ByVal objButton As Control, ByVal objContent As Control,
ByVal intModuleId As Integer, ByVal blnDefaultMin As Boolean, ByVal strMinIconLoc As
String, ByVal strMaxIconLoc As String, ByVal ePersistanceType As MinMaxPersistanceType)
```

Parameter	Description
`objButton`	Control that when clicked causes content area to be hidden/shown
`objContent`	Content area that is hidden/shown
`intModuleId`	Module id of button/content, used only for persistance type of Cookie
`blnDefaultMin`	If content area is to be defaulted to minimized pass in true
`strMinIconLoc`	Location of minimized icon
`strMaxIconLoc`	Location of maximized icon
`ePersistanceType`	How to store current state of min/max. Cookie, Page, None

There is also the need for the modules to know if the current state of the content is visible or not. This has also been abstracted into a new function called `MinMaxContentVisibile`.

```
Public Shared Property MinMaxContentVisibile(ByVal objButton As Control,
ByVal intModuleId As Integer, ByVal blnDefaultMin As Boolean,
ByVal ePersistanceType As MinMaxPersistanceType) As Boolean
```

Parameter	Description
`objButton`	Control that when clicked causes content area to be

	hidden/shown
intModuleId	Module id of button/content, used only for persistance type of Cookie
blnDefaultMin	If content area is to be defaulted to minimized pass in true
ePersistanceType	How to store current state of min/max. Cookie, Page, None

The following modifications were made to incorporate the use of these new methods.

HelpButtonControl - EnableMinMax

```
DotNetNuke.UI.Utilities.DNNClientAPI.EnableMinMax(cmdHelp, pnlHelp, True, _
Utilities.DNNClientAPI.MinMaxPersistanceType.None)
```

SectionHeadControl - EnableMinMax

```
DotNetNuke.UI.Utilities.DNNClientAPI.EnableMinMax(imgIcon, ctl, Not IsExpanded,
Page.ResolveUrl(MinImageUrl), Page.ResolveUrl(MaxImageUrl), _
Utilities.DNNClientAPI.MinMaxPersistanceType.Page)
```

Visibility - EnableMinMax

```
If (PortalModule.ModuleConfiguration.Visibility = VisibilityState.Maximized) OrElse _
(PortalModule.ModuleConfiguration.Visibility = VisibilityState.Minimized) Then
  DNNClientAPI.EnableMinMax(cmdVisibility, ModuleContent, PortalModule.ModuleId, _
PortalModule.ModuleConfiguration.Visibility = VisibilityState.Minimized,
MinIconLoc, MaxIconLoc, DNNClientAPI.MinMaxPersistanceType.Cookie)
End If
```

HelpButtonControl – MinMaxContentVisible

N/A

SectionHeadControl - MinMaxContentVisible

```
Public Property IsExpanded() As Boolean
  Get
    Return DotNetNuke.UI.Utilities.DNNClientAPI.MinMaxContentVisibile(imgIcon, Not
_isExpanded, Utilities.DNNClientAPI.MinMaxPersistanceType.Page)
  End Get
  Set(ByVal Value As Boolean)
     isExpanded = Value
    DotNetNuke.UI.Utilities.DNNClientAPI.MinMaxContentVisibile(imgIcon, Not _isExpanded,
Utilities.DNNClientAPI.MinMaxPersistanceType.Page) = Value
  End Set
End Property
```

Visibility – MinMaxContentVisible

```
Public Property ContentVisible() As Boolean
```

```
  Get
    Select Case PortalModule.ModuleConfiguration.Visibility
      Case Entities.Modules.VisibilityState.Maximized,
Entities.Modules.VisibilityState.Minimized
        Return DNNClientAPI.MinMaxContentVisibile(cmdVisibility, PortalModule.ModuleId,
PortalModule.ModuleConfiguration.Visibility = Entities.Modules.VisibilityState.Minimized,
DNNClientAPI.MinMaxPersistanceType.Cookie)
      Case Else
        Return True
    End Select
  End Get
  Set(ByVal Value As Boolean)
    DNNClientAPI.MinMaxContentVisibile(cmdVisibility, PortalModule.ModuleId, _
PortalModule.ModuleConfiguration.Visibility = Entities.Modules.VisibilityState.Minimized,
DNNClientAPI.MinMaxPersistanceType.Cookie) = Value
  End Set
End Property
```

Showing and Hiding Module Content

The task of showing and hiding content in a browser that supports DHTML is rather simple. Simply get a reference to the element and access its style.display property and set it to none or empty-string depending on whether you wish to show or hide the element. The event that triggers this action is clicking on an image with a plus to maximize (show) content and clicking on a minus image to minimize (hide) the content. Finally, the way the module remembers its state (whether it was maximized or minimized) is by using a cookie unique to the module.

Handing the Click Event
The functionality behind the showing and hiding of modules is found in the Visibility.ascx UserControl. The control's job was to check for a cookie to determine if it should show or hide content and set the appropriate image (plus/minus, or custom). Additionally, it set the visibility of the content.

This enhancement first performs a check to see if the browser supports basic DHTML by the following code.

```
If ClientAPI.BrowserSupportsFunctionality(ClientAPI.ClientFunctionality.DHTML) Then
```

If this returns True then we do the following

Always set the content visibility to True

Since the content needs to be available on the client side we always need it to be rendered.

When content is supposed to be hidden set the content's display style to none

Register the Client API for the dnn.dom namespace

```
ClientAPI.RegisterClientReference(Page, ClientAPI.ClientNamespaceReferences.dnn_dom)
```

Add a client-side onclick handler to the image

Setting an event handler via code like this

```
cmdVisibility.Attributes.Add("onclick", "if (__dnn_ModuleMaxMin_OnClick(this, '" & _
pnlModuleContent.ClientID & "')) return false;")
```

This will cause this event to fire before the postback code is run. If we return false, then the postback code will never run. This is exactly what we want, if the max/min script succeeds we do not want a postback to occur, however, if it fails we will fall back on the reliable postback handling.

Set a custom attribute on the button to denote the moduleid

Since the client-side will be responsible for setting the cookie to maintain the module's max/min state, we need to know the module's id in order to set the correct cookie.

Register 2 client-side variables for the max and min image locations

A module can have a custom set of images assigned to its max and min button. Since the client is responsible for switching the image when clicked we need to know what image to use. This information is passed down to the client by using the `ClientAPI.RegisterClientVariable` method. On the client-side it is retrieved with the `dnn.getVar` method.

Fixing the Enter Key for Account Login

This enhancement also fixed the issue we were having for the Account Login module. When the user pressed the enter key and the max/min button was shown, the max/min button would receive the action instead of the login button. This is because the Max/Min button used to be of type ImageButton, which would render the HTML <INPUT TYPE="image". According to the HTML spec, this element acts in the same way as a submit button. Thus, it would receive the action for the Enter key. I have worked around this issue by converting the Visibility buttons from ImageButtons to a combination of an Image and a LinkButton.

Additional Information

The DotNetNuke Portal Application Framework is constantly being revised and improved. To ensure that you have the most recent version of the software and this document, please visit the DotNetNuke website at:

http://www.dotnetnuke.com

The following additional websites provide helpful information about technologies and concepts related to DotNetNuke:

DotNetNuke Community Forums
http://www.dotnetnuke.com/tabid/795/Default.aspx

Microsoft® ASP.Net
http://www.asp.net

Open Source
http://www.opensource.org/

W3C Cascading Style Sheets, level 1
http://www.w3.org/TR/CSS1

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	Aug 16, 2005	Shaun Walker	• Applied new template