DotNetNuke Wizard Framework

Charles Nurse

Version 1.0.0
Last Updated: June 20, 2006
Category: Wizard

DotNetNuke Wizard Framework

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.

DotNetNuke Wizard Framework

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 Wizard Framework 1

Introduction 1

Wizards 2

Wizard Framework 3

The Wizard Class 4

Wizard API Reference 6

Additional Information 9

Appendix A: Document History 10

DotNetNuke Wizard Framework

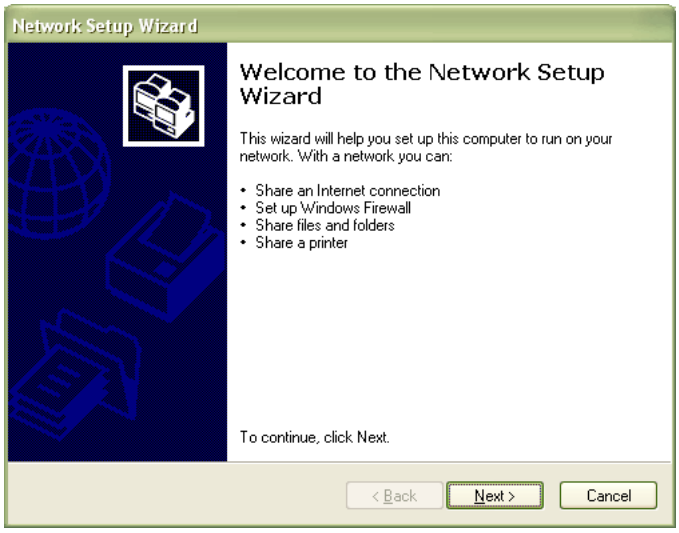
Introduction

DotNetNuke has grown in size considerably over the last few months. As a result, the large number of properties that often need to be set up can intimidate many neophyte users. Most users before version 2.0 were developers looking for a framework to build their own sites. As the discussions on the ASP.Net Forums indicate, more and more users are not primarily developers but are just simply looking for a hosting solution they can use.

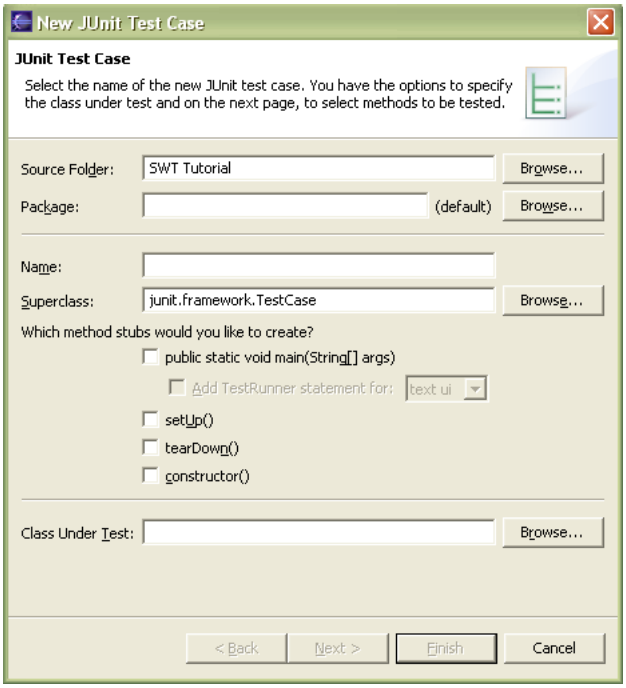
One mechanism to help novice users is to ensure that adequate Help is incorporated into an Application. Another mechanism is to provide "Wizards" that guide the user through a series of steps.

Wizards

There are many styles of Wizard in common use today. Below is an example of the style implemented by Microsoft. This particular example is from the Network SetUp Wizard in Windows XP. In this style of Wizard, the left hand side of the Wizard contains an image, the right hand side contains the text and controls used for user input. On the bottom, is a Button Panel, that is used for navigation through the pages.



A second type of Wizard in common use, is exemplified by the following example – "New JUnit Test Case" from Eclipse 3.0. This type of Wizard is designed for a more a technically inclined user group, so there are usually more user options on one page, together with a title and a short introduction in the Header panel. However, this type of Wizard still includes the Next/Back/Finish/Cancel button panel on the bottom of the page.



Wizard Framework

One of DotNetNuke's strengths are the tools it provides to other developers. When the DotNetNuke team decided that a Site Wizard would probably simplify the user experience in configuring a site, it was felt that other developers could probably take advantage of a Wizard API or Framework.

The Wizard Framework (or API) consists of several new Classes found in the components/Wizards folder. These are

- WizardPage.vb This class defines a Wizard Page, and contains 4 Public Properties:
  - Icon A Url to an image about 36 x 36 that will display in the top left of the page
  - Title The title of the page that will appear in the Header of the Wizard
  - Control The control that displays in the body of the Wizard
  - Type The type of Wizard Page – This is an Enum, valid values being Content, Success and Failure
- WizardPageCollection.vbThis class subclasses CollectionBase to provide a type-safe collection of Wizard Pages.
- WizardEventArgs.vb & WizardCancelEventArgs.vb These two classes subclass EventArgs to provide Wizard specific EventArgs properties for the Events raised by Wizard.vb
- Wizard.vbThe Wizard class itself subclasses PortalModuleControl, and this is the class that developers need to inherit from, when the develop there own Wizards (more below)
- Success.ascx This user control is a default Success/Failure page for the Wizard Framework. Developers can develop theire own Success/Failure pages or they can use this default class

The Wizard Class

Wizard.vb contains the Wizard Class that is the core of the Framework. This class handles most of the repetitive tasks that a wizard needs to function, and exposes Methods, Properties and Events that a specific Wizard can use, as well as a number of Private Helper Methods hat implement the common tasks.

Wizard subclasses PortalModuleControl, so it has access to the same functionality as any DotNetNuke module. As it therefore inherits indirectly from UserControl, its Page Init Event Handler is used to add the Header (Image and Title) Row to the Wizard, as well as adding the Button Panel in the Footer. To accomplish this, any class that inherits from Wizard must provide the following basic structure, as part of its ascx file.

```
<table id="Wizard" runat="server">
   <tr>
      <td id="WizardBody" valign="top">
         <asp:panel id="pnlPage1" runat="server">
         …….
         </asp:panel>
         <asp:panel id="pnlPage2" runat="server">
         …….
         </asp:panel>
         …….
      </td>
   </tr>
</table>
```

The table element “Wizard” and the tablecell element “WizardBody” must exist with the names specified. This is so that the Page.Init event in Wizard can locate where to inject the Wizard Header Row and Footer Row.

The panels represent the pages of the control. These pages need not be panels – they can be any Web Control - and they do not require a specific naming convention.

In order to add pages to the Wizard, The Page Load event Handler of the subclass can make calls to the Wizard’s AddPage method.

```
Private Sub Page_Load(….) Handles MyBase.Load
   AddPage("Introduction", "~/images/icon1.gif", pnlPage1)
   AddPage("Next", "~/images/icon2.gif", pnlPage2)

   --- Add Other Pages --

If Not Page.IsPostBack Then
--- Insert custom user code ---

      CurrentPage = 0  'Set CurrentPage to the first Page
      FinishPage = 2          'Set FinishPage to 2 (third page)
      DisplayCurrentPage()    'Display First (Current page)
   End If
End Sub
```

The first parameter of the AddPage method is the title (and should probably be localized using the Localization API).

The second parameter is the Url of the icon to be displayed in the top left of the Wizard, and the third parameter is the control that will be displayed in the WizardBody for the page.

The current page is then set to 0 (so that DisplayCurrentPage displays the first page – as most .NET collections the WizardPageCollection class is 0 based). The finish page is set to 2 (the third page). This is the first page where the finish button is enabled.

The subclass can respond to three events raised by the base class.

- BeforePageChanged Can be used to check to validate before proceeding to the next page, as the WizardCancelEventArgs object exposes a Cancel property, that if set will roll back the proposed Page Change
- AfterPageChanged Can be used to carry out logic that must happen between pages.
- FinishWizard Can be used to carry out logic that must happen to complete the Wizard tasks.

In most cases, the "Update" logic will all happen when the Finish button is complete, but there is nothing in the Framework that prevents the users choices being committed after each page.

Wizard API Reference

Properties

CurrentPage An Integer that sets or retrieves the current page (Note – setting the currentpage is not sufficient to display the page. A call to DisplayCurrentPage must be made as well.)

FinishPage An Integer that sets or retrieves the first page in the Wizard where the "Finish" button is enabled. This is used to force the user to proceed through some pages before the Wizard can finish.

FailurePage Sets or retrieves a WizardPage object that represents the Page to display if the Wizard fails when the Finish button is cliked

Pages A collection of WizardPage objects that represent the content pages of the Wizard.

SuccessPage Sets or retrieves a WizardPage object that represents the Page to display if the Wizard succeeds when the Finish button is clicked

Methods

AddPage(title,icon,ctl) & AddPage(title,icon,ctl,type) Two overloads of AddPage that Add a page to the Wizard. The first overload creates a "ContentPage" and adds it to the Pages collection. The second overload creates a specific type of page. A "Content" page is added to the Pages collection as in the first overload, A "Success" page will set the SuccessPage Property, and a "Failure" page will set the FailurePage property

Title: the title of the Page
Icon: the icon displayed in the top left corner
ctl: the control that displays in the Wizard Body
type: the type of Control (an enumeration)
- WizardPageType.Content
- WizardPageType.Failure
- WizardPageType.Success

DisplayCurrentPage() Displays the currentpage (as determined by the CurrentPage property)

DisplayFailurePage(message) Displays the failure page, either the default failure page or a user provided page (if the FailurePage property is set).

message: a message that can be added to the default failure page (it will be displayed in red)

DisplayPage(pageNo,wizPage,ShowBack, ShowNext, ShowFinish)
Displays a specific page in the Wizard Body.

This method allows the subclass to override the normal behaviour of the Wizard by displaying pages out of sequence.

pageNo: The page no of the page to display
wizPage: The WizardPage object to display
ShowBack: A flag to set whether the Back button displays
ShowNext: A flag to set whether the Next button displays
ShowFinish: A flag to set whether the Finish button displays

DisplaySuccessPage(message) Displays the success page, either the default success page or a user provided page (if the SuccessPage property is set).

message: a message that can be added to the default success

EnableCommand(cmd, enable) This method allows buttons to be enabled/disabled, over-riding the normal behaviour of the wizard

cmd: the button to override. This is an enumeration the options are:
WizardCommand.Cancel
WizardCommand.Finish
WizardCommand.PreviousPage
WizardCommand.NextPage

Events

AfterPageCanged(sender, wizardeventargs) Fires after a page has changed. The WizardEventArgs object has four properties:

Page: The WizardPage that is now the current page
PageNo: The page no of the WizardPage that is now the current page
PreviousPage: The WizardPage that was previously current
PreviousPageNo: The page no of the previous WizardPage object

BeforePageChanged(sender, wizardcanceleventargs) Fires before a page has changed. The WizardCancelEventArgs object has five properties. It subclasses WizardEventArgs, with the addition of a cancel property, that is always false when the event is fired. Setting this property to True, des not allow the page change to complete, rolling back any changes.

FinishWizard(sender, wizardeventargs) Fires when the Finish button is clicked this event allows the subclass to run any logic required to complete the Wizard and optionally display a success or failure page.

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