DotNetNuke Search Engine

Shaun Walker

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

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 Search Engine Architecture

Search Engine is called by the Host / Search administrator UI or by the background scheduled job (DotNetNuke.Services.Search.SearchEngineScheduler)

```
Dim se As New Services.Search.SearchEngine
se.IndexContent()
```

SearchEngine utilizes 2 independent providers:
The IndexingProvider is responsible for getting the items to index from the modules (also known as a Crawler)
The SearchDataStoreProvider is responsible for processing the items and storing them in a persistent storage format (also known as a Parser)

SearchEngine.vb

```
Public Sub IndexContent()
        Dim Indexer As IndexingProvider = IndexingProvider.Instance


SearchDataStoreProvider.Instance.StoreSearchItems(GetContent(Indexer))
End Sub
```

GetContent is a helper method which iterates through all portals and calls GetSearchIndexItems

```
    Protected Function GetContent(ByVal Indexer As IndexingProvider) As
SearchItemInfoCollection
        Dim SearchItems As New SearchItemInfoCollection
        Dim objPortals As New PortalController
        Dim objPortal As PortalInfo

        Dim arrPortals As ArrayList = objPortals.GetPortals
```

```
        Dim intPortal As Integer
        For intPortal = 0 To arrPortals.Count - 1
            objPortal = CType(arrPortals(intPortal), PortalInfo)

            SearchItems.AddRange(Indexer.GetSearchIndexItems(objPortal.PortalID))

        Next
        Return SearchItems
    End Function
```

The web.config defines where to locate the default IndexingProvider and SearchDataStoreProvider implementations

web.config

```
    <section name="searchIndex"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"
/>
    <section name="searchDataStore"
type="DotNetNuke.Framework.Providers.ProviderConfigurationHandler, DotNetNuke"
/>
    ...
    <searchIndex defaultProvider="ModuleIndexProvider">
      <providers>
        <clear />
        <add name="ModuleIndexProvider"
type="DotNetNuke.Services.Search.ModuleIndexer, DotNetNuke.Search.Index"
providerPath="~\Providers\SearchProviders\ModuleIndexer\" />
      </providers>
    </searchIndex>
    <searchDataStore defaultProvider="SearchDataStoreProvider">
      <providers>
        <clear />
        <add name="SearchDataStoreProvider"
type="DotNetNuke.Services.Search.SearchDataStore, DotNetNuke.Search.DataStore"
providerPath="~\Providers\SearchProviders\SearchDataStore\" />
      </providers>
    </searchDataStore>
```

IndexingProvider defines the abstract class

IndexingProvider.vb

```
    Public MustOverride Function GetSearchIndexItems(ByVal PortalID As Integer) As
SearchItemInfoCollection
```

ModuleIndexer provides the implementation for the IndexingProvider abstract class

ModuleIndexer.vb

Calls the modules GetSearchItems method and creates a collection of SearchItemInfo objects

```
    Public Overrides Function GetSearchIndexItems(ByVal PortalID As Integer) As
SearchItemInfoCollection

        Dim SearchItems As New SearchItemInfoCollection
        Dim SearchCollection As SearchContentModuleInfoCollection =
GetModuleList(PortalID)

        For Each ScModInfo As SearchContentModuleInfo In SearchCollection
            Try
                Dim myCollection As SearchItemInfoCollection
                myCollection =
ScModInfo.ModControllerType.GetSearchItems(ScModInfo.ModInfo)
                If Not myCollection Is Nothing Then
                    SearchItems.AddRange(myCollection)
                End If
            Catch ex As Exception
                LogException(ex)
            End Try
        Next

        Return SearchItems

    End Function
```

Helper method to get list of modules which implement ISearchable

```
    Protected Function GetModuleList(ByVal PortalID As Integer) As
SearchContentModuleInfoCollection

        Dim Results As New SearchContentModuleInfoCollection

        Dim objModules As New ModuleController
        Dim arrModules As ArrayList = objModules.GetSearchModules(PortalID)
        Dim businessControllers As New Hashtable
        Dim htModules As New Hashtable

        Dim objModule As ModuleInfo
```

```
        For Each objModule In arrModules
            If Not htModules.ContainsKey(objModule.ModuleID) Then
                Try
                    'Check if the business controller is in the Hashtable
                    Dim objController As Object =
businessControllers(objModule.BusinessControllerClass)

                    'If nothing create a new instance
                    If objController Is Nothing Then
                        objController =
Framework.Reflection.CreateObject(objModule.BusinessControllerClass,
objModule.BusinessControllerClass)

                        'Add to hashtable
                        businessControllers.Add(objModule.BusinessControllerClass,
objController)
                    End If

                    'Double-Check that module supports ISearchable
                    If TypeOf objController Is ISearchable Then
                        Dim ContentInfo As New SearchContentModuleInfo
                        ContentInfo.ModControllerType = CType(objController, ISearchable)
                        ContentInfo.ModInfo = objModule
                        Results.Add(ContentInfo)
                    End If
                Catch ex As Exception
                    LogException(ex)
                Finally
                    htModules.Add(objModule.ModuleID, objModule.ModuleID)
                End Try
            End If
        Next

        Return Results

    End Function
```

Modules implement the ISearchable interface - GetSearchItems method

```
    Implements Entities.Modules.ISearchable

    Public Function GetSearchItems(ByVal ModInfo As Entities.Modules.ModuleInfo) As
Services.Search.SearchItemInfoCollection Implements
Entities.Modules.ISearchable.GetSearchItems
        Dim SearchItemCollection As New SearchItemInfoCollection
```

```
        Dim Announcements As ArrayList = GetAnnouncements(ModInfo.ModuleID)

        Dim objAnnouncement As Object
        For Each objAnnouncement In Announcements
            Dim SearchItem As SearchItemInfo
            With CType(objAnnouncement, AnnouncementInfo)
                Dim UserId As Integer = Null.NullInteger
                If IsNumeric(.CreatedByUser) Then
                    UserId = Integer.Parse(.CreatedByUser)
                End If

                Dim strContent As String = System.Web.HttpUtility.HtmlDecode(.Title & "
" & .Description)
                Dim strDescription As String =
HtmlUtils.Shorten(HtmlUtils.Clean(System.Web.HttpUtility.HtmlDecode(.Description),
False), 100, "...")

                SearchItem = New SearchItemInfo(ModInfo.ModuleTitle & " - " & .Title,
strDescription, UserId, .CreatedDate, ModInfo.ModuleID, .ItemId.ToString, strContent,
"ItemId=" & .ItemId.ToString)
                SearchItemCollection.Add(SearchItem)
            End With
        Next

        Return SearchItemCollection
    End Function
```

SearchDataStoreProvider defines the abstract class for saving and retrieving from the search data store

SearchDataStoreProvider.vb

```
    Public MustOverride Sub StoreSearchItems(ByVal SearchItems As
SearchItemInfoCollection)
    Public MustOverride Function GetSearchResults(ByVal PortalID As Integer, ByVal
Criteria As String) As SearchResultsInfoCollection
    Public MustOverride Function GetSearchItems(ByVal PortalID As Integer, ByVal
TabID As Integer, ByVal ModuleID As Integer) As SearchResultsInfoCollection
```

SearchDataStore processes the search items and stores them in a persistent storage location.
The default SearchDataStore performs all relevancy filtering and splits the content into keywords

which are stored in an inverted list for fast retrieval.
SearchDataStore also provides the implementation for retrieving items from the data store.

SearchDataStore.vb

```
    Public Overrides Sub StoreSearchItems(ByVal SearchItems As
SearchItemInfoCollection)

        Dim i As Integer

        'Get the default Search Settings
        _defaultSettings = Common.Globals.HostSettings


        'For now as we don't support Localized content - set the locale to the default
locale. This
        'is to avoid the error in GetDefaultLanguageByModule which artificially limits
the number
        'of modules that can be indexed.  This will need to be addressed when we
support localized content.
        Dim Modules As New Hashtable
        For i = 0 To SearchItems.Count - 1
            If Not Modules.ContainsKey(SearchItems(i).ModuleId.ToString) Then
                Modules.Add(SearchItems(i).ModuleId.ToString, "en-US")
            End If
        Next

        Dim SearchItem As SearchItemInfo
        Dim IndexedItem As SearchItemInfo
        Dim IndexedItems As SearchItemInfoCollection
        Dim ModuleItems As SearchItemInfoCollection
        Dim IndexID As Integer
        Dim iSearch As Integer
        Dim ModuleId As Integer
        Dim Language As String
        Dim ItemFound As Boolean

        'Process the SearchItems by Module to reduce Database hits
        Dim moduleEnumerator As IDictionaryEnumerator = Modules.GetEnumerator()
        While moduleEnumerator.MoveNext()
            ModuleId = CType(moduleEnumerator.Key, Integer)
            Language = CType(moduleEnumerator.Value, String)

            'Get the Indexed Items that are in the Database for this Module
```

```
            IndexedItems = GetSearchItems(ModuleId)
            'Get the Module's SearchItems to compare
            ModuleItems = SearchItems.ModuleItems(ModuleId)

            'As we will be potentially removing items from the collection iterate
backwards
            For iSearch = ModuleItems.Count - 1 To 0 Step -1
                SearchItem = ModuleItems(iSearch)
                ItemFound = False

                'Iterate through Indexed Items
                For Each IndexedItem In IndexedItems
                    'Compare the SearchKeys
                    If SearchItem.SearchKey = IndexedItem.SearchKey Then
                        'Item exists so compare Dates to see if modified
                        If IndexedItem.PubDate < SearchItem.PubDate Then
                            Try
                                'Content modified so update SearchItem and delete item's
Words Collection
                                SearchItem.SearchItemId = IndexedItem.SearchItemId
                                SearchDataStoreController.UpdateSearchItem(SearchItem)

SearchDataStoreController.DeleteSearchItemWords(SearchItem.SearchItemId)

                                ' re-index the content
                                AddIndexWords(SearchItem.SearchItemId, SearchItem,
Language)
                            Catch ex As Exception
                                'Log Exception
                                LogException(ex)
                            End Try
                        End If

                        'Remove Items from both collections
                        IndexedItems.Remove(IndexedItem)
                        ModuleItems.Remove(SearchItem)

                        'Exit the Iteration as Match found
                        ItemFound = True
                        Exit For
                    End If
                Next

                If Not ItemFound Then
                    Try
```

```
                    'Item doesn't exist so Add to Index
                    IndexID = SearchDataStoreController.AddSearchItem(SearchItem)
                    ' index the content
                    AddIndexWords(IndexID, SearchItem, Language)
                Catch ex As Exception
                    'Log Exception
                    LogException(ex)
                End Try
            End If

        Next

        'As we removed the IndexedItems as we matched them the remaining items
are deleted Items
        'ie they have been indexed but are no longer present
        Dim ht As New Hashtable
        For Each IndexedItem In IndexedItems
            Try
                'dedupe
                If ht(IndexedItem.SearchItemId) Is Nothing Then

SearchDataStoreController.DeleteSearchItem(IndexedItem.SearchItemId)
                    ht.Add(IndexedItem.SearchItemId, 0)
                End If
            Catch ex As Exception
                'Log Exception
                LogException(ex)
            End Try
        Next

    End While

End Sub

Public Overloads Overrides Function GetSearchItems(ByVal PortalID As Integer,
ByVal TabID As Integer, ByVal ModuleID As Integer) As SearchResultsInfoCollection
    Return New
SearchResultsInfoCollection(CBO.FillCollection(Data.DataProvider.Instance().GetSearc
hItems(PortalID, TabID, ModuleID), GetType(SearchResultsInfo)))
End Function

Public Overrides Function GetSearchResults(ByVal PortalID As Integer, ByVal
Criteria As String) As SearchResultsInfoCollection

    'We will assume that the content is in the locale of the Portal
```

```
        Dim objPortalController As New PortalController
        Dim objPortal As PortalInfo = objPortalController.GetPortal(PortalID)
        Dim locale As String = objPortal.DefaultLanguage
        Dim CommonWords As Hashtable = GetCommonWords(locale)

        ' clean criteria
        Criteria = Criteria.ToLower

        ' split search criteria into words
        Dim SearchWords As New SearchCriteriaCollection(Criteria)

        Dim SearchResults As New Hashtable
        ' iterate through search criteria words
        Dim Criterion As SearchCriteria
        For Each Criterion In SearchWords
            If CommonWords.ContainsKey(Criterion.Criteria) = False Then
                Dim ResultsCollection As SearchResultsInfoCollection =
SearchDataStoreController.GetSearchResults(PortalID, Criterion.Criteria)
                If Criterion.MustExclude = False Then
                    ' Add all these to the results
                    For Each Result As SearchResultsInfo In ResultsCollection
                        If SearchResults.ContainsKey(Result.SearchItemID) Then
                            CType(SearchResults.Item(Result.SearchItemID),
SearchResultsInfo).Relevance += Result.Relevance
                        Else
                            SearchResults.Add(Result.SearchItemID, Result)
                        End If
                    Next
                End If
                If Criterion.MustInclude Then
                    ' We need to remove items which do not include this term
                    Dim MandatoryResults As New Hashtable
                    For Each result As SearchResultsInfo In ResultsCollection
                        MandatoryResults.Add(result.SearchItemID, 0)
                    Next
                    For Each Result As SearchResultsInfo In SearchResults.Values
                        If MandatoryResults.ContainsKey(result.SearchItemID) = False Then
                            result.Delete = True
                        End If
                    Next
                End If
                If Criterion.MustExclude Then
                    ' We need to remove items which do include this term
                    Dim ExcludedResults As New Hashtable
                    For Each result As SearchResultsInfo In ResultsCollection
```

```
                    ExcludedResults.Add(result.SearchItemID, 0)
                Next
                For Each Result As SearchResultsInfo In SearchResults.Values
                    If ExcludedResults.ContainsKey(result.SearchItemID) = True Then
                        Result.Delete = True
                    End If
                Next
            End If
        End If
    Next

    'Only include results we have permission to see
    Dim Results As New SearchResultsInfoCollection
    For Each SearchResult As SearchResultsInfo In SearchResults.Values

        'Check If authorised to View Tab
        Dim objTabController As New TabController
        Dim objTab As TabInfo = objTabController.GetTab(SearchResult.TabId)
        If PortalSecurity.IsInRoles(objTab.AuthorizedRoles) Then

            'Now check if authorized to view module
            Dim objModuleController As New ModuleController
            Dim objModule As ModuleInfo =
objModuleController.GetModule(SearchResult.ModuleId, SearchResult.TabId)
            If PortalSecurity.IsInRoles(objModule.AuthorizedViewRoles) = True And
objModule.IsDeleted = False Then
                'If authorised add result to collection
                Results.Add(SearchResult)
            End If
        End If
    Next


    'Return Search Results Collection
    Return Results
End Function
```

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