Thesaurus Desktop

v1.1 .NET Edition

Software Component

User's Guide

Keyoti

Contents

Terms & Conditions

If you do not agree to these terms and conditions then you may not install, use, or distribute Thesaurus Desktop.

1. License and Ownership. Thesaurus Desktop is a licensed software product. Thesaurus Desktop, including its accompanying files and documentation, is owned and copyrighted by Keyoti Inc., © Copyright 2004-2008, all rights reserved. Keyoti Inc. grants the user in possession of an official receipt of purchase a nonexclusive license to download and use the software, for any lawful commercial or non-commercial purposes provided the terms of this agreement are met.
2. Distribution. If you purchased an Individual License Version of Thesaurus Desktop, you may copy and use the Thesaurus Desktop distribution file (.zip) on any systems you use but for your individual use only. Individual use is defined as one person having the ability to read or copy any of the files originally contained in the Thesaurus Desktop distribution file including (but not limited to) the .dll, .cs, .vb, .html, .gif, .jpg, .bat, and .txt files. You must take appropriate safeguards to protect this product from unauthorized access by others.
If you purchased a Site License Version of Thesaurus Desktop, then you may provide copies of the Thesaurus Desktop distribution file (.zip) to anyone employed by your company whose primary work address is within a 100 mile (160 km) radius of your primary work address. These individuals then have license rights and responsibilities equivalent to the Individual License Version. In all cases the copies of the Thesaurus Desktop distribution file (.zip) must be unaltered and complete, including this license agreement and all copyright notices.
You may not reverse engineer, disassemble, decompile, or modify this software or its accompanying documentation in any way.
You may reproduce and redistribute a copy of the dll files as part of any .NET based software developed and licensed by you that itself uses Thesaurus Desktop provided
a) your software has clearly distinct and added functionality,
b) you are responsible for all technical support,
c) the Thesaurus Desktop application programming interfaces are not documented, exposed, or otherwise made available by your software,
d) attribution is given to Keyoti Inc. (http://www.keyoti.com) in any documentation or screen displays where other credits appear,
e) you do not allow recipients of your software to reverse engineer, disassemble, decompile, or copy portions from your software allowing them to gain separate access to Thesaurus Desktop or any parts of it,
f) your software is not used by software developers as part of their application, and
g) your software is not a software component or 'Control'.
The source code samples included with this package and in the Thesaurus Desktop documentation may be freely used, modified, incorporated, and distributed without restriction as part of any software that uses Thesaurus Desktop itself.
3. Warranty Disclaimer and Limitation of Liability. Thesaurus Desktop is licensed to the user on an "AS IS" basis. Keyoti Inc. MAKES NO WARRANTIES, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS SOFTWARE AND ITS ASSOCIATED FILES AND DOCUMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. KEYOTI Inc. DOES NOT WARRANT THAT THE OPERATION OF THIS SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT DEFECTS WILL BE CORRECTED. You the user are solely responsible for determining the appropriateness of this software for your use and accept full responsibility for all risks associated with its use. Keyoti Inc. is not and will not be liable for any direct, indirect, special or incidental damages in any amount including loss of profits or interruption of business however caused, even if Keyoti Inc. has been advised of the possibility of such damages. Keyoti Inc. retains the right to, in its sole discretion, refund the purchase price of this software as a complete and final resolution to any dispute.
SPECIFIC DISCLAIMER FOR HIGH-RISK ACTIVITIES.
The SOFTWARE is not designed or intended for use in high-risk activities including, without restricting the generality of the foregoing, on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Keyoti Inc. and its suppliers specifically disclaim any express or implied warranty of fitness for such purposes or any other purposes.
NO LIABILITY FOR CONSEQUENTIAL DAMAGES.
To the maximum extent permitted by applicable laws, in no event shall Keyoti Inc. or its suppliers be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of information, or other pecuniary loss) arising out of the use of or inability to use this Keyoti Inc. product, even if Keyoti Inc. has been advised of the possibility of such damages. Because some states and jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.
4. Support and Source Code Licensing. Keyoti Inc. does not provide formal customer support for Thesaurus Desktop. However, we make every effort to ensure the quality of our products, and questions, bug reports, and feature requests are encouraged, please use email. Due to resource limitations, it may not be possible to resolve all issues and no assurances can be given as to when or if problems will be addressed. Customers incorporating Thesaurus Desktop into software for deployment or sale may purchase the Source Code License Version of Thesaurus Desktop which will enable self-support, debugging, and modifications or extensions to the base functionality (but not distribution of the source code itself).

Introduction

Thank you for purchasing Thesaurus from Keyoti, please keep up to date through our website keyoti.com and feel invited to post feedback. We also ask that you register your purchase so that we may offer you support and free upgrades from time to time, at http://keyoti.com/care/register

Overview

Thesaurus consists of 3 components designed for a multitude of uses; from simple out-of-the box dialog and context menu operation (similar to popular word processors), through customized user interface and behavior and on to non GUI SDK usage.

Specification

Contains over 60,000 English terms
1 Thesaurus model/logic component
1 Windows Control + 1 Windows Component
Dialog based Thesaurus with meanings and their synonyms/antonyms
Context Menu based synonyms
Fully customizable UI
Open API/SDK
Compatible with all .NET text boxes and most 3rd party boxes

Requirements

The components were built to .NETSDK 1.0, 1.1, 2.0, 3.0, 3.5 specification and are designed to work with all SDKs from 1.0 up. The components and controls are housed in one Windows Forms DLL (<100KB) and one 'model' DLL which contains the actual thesaurus (~3.5MB).

Installation

Running the MSI installer is all that is required to install the controls for development. Initially the controls will be licensed on a time-limited trial basis until a license key is registered through the evaluation splash page.

After installation, the DLLs are added to the GAC and made available to the list of assemblies in Visual Studio, from here the controls can be added to the VS toolbar by right clicking in the Toolbox and choosing "Add/Remove items..." and from there selecting the controls in the dialog list.

Licensing

The Thesaurus controls, components and classes are licensed using the .NET licensing framework. Fundamental to this is the requirement that a file "licenses.licx" is compiled with the project.

Registering The License Key

Either click "Register" in the blue licensing dialog shown when running an unlicensed application, or run the registration application under Start->Thesaurus Desktop .NET->Register License Key. Any applications using Thesaurus must subsequently be fully rebuilt (Rebuild Solution) to embed the key properly.

Visual Studio

A licenses.licx file is created (or appended to) when any licensed control is dragged onto the design surface. However, if the controls are used directly in code, or if the ThesaurusEngine class is used directly then Visual Studio will not create/append the licenses.licx file. In this case create or open one in the project directory (using Notepad for example - make sure it is called "licenses.licx" and not "licenses.licx.txt"). In the licenses.licx file add the line:

```
Keyoti.Thesaurus.Model.ThesaurusEngine, Keyoti.Thesaurus.Model.NET2
```

which follows the standard format

```
<Class name>, <Assembly name>
```

After modifyin the licenses.licx file it is important to **rebuild** the **solution**.

Note: if a licenses.licx file is copied into the project directory, it must also be added to the VS project.

Command-line compiler

lc.exe is used to compile a text file (like licenses.licx) into a resource, from there the resource can be compiled into the final executable, please see the .NET documentation and Non Visual Studio examples for help.

The Controls And Components

Keyoti.Thesaurus.Windows.Thesaurus

The "Thesaurus" component is housed in the Keyoti.Thesaurus.Windows namespace, in the assembly of the same name. It's pupose is to act as a 'controller' between the view (Keyoti.Thesaurus.Windows.ThesaurusControl) and the model (Keyoti.Thesaurus.Model.ThesaurusEngine). It is typically added to Windows Forms to provide the developer with convenient access to the thesaurus dialog, and context menu items.

Commonly Used Methods

Show - Shows the thesaurus dialog, non-modally, working with the word nearest the cursor.

ShowDialog - Shows the thesaurus dialog, modally, working with the word nearest the cursor.

GetSuggestionMenuItems - Returns MenuItemCollection of suggestions (synonyms, antonyms or related words) if there are any, for word that was clicked on or Apps key activated on.

Keyoti.Thesaurus.Windows.ThesaurusControl

Is a simple implementation of IThesaurusControl. It is a UserControl with no logic, only the user interface elements required for the dialog. This control can be tied to the "Thesaurus" control above through the Thesaurus.ThesaurusControl property (or via a Thesaurus.ThesaurusControlProvider). Note the source code for this control is also included in the custom layout examples.

Keyoti.Thesaurus.Model.ThesaurusEngine

Contains the core functionality of the Thesaurus and exposes methods which can be used by other classes. This class can be tied to the "Thesaurus" control above through the Thesaurus.ThesaurusEngine property.

Commonly Used Methods

GetRelatedWords - Returns list of words that share the same stem (lemmas).

GetAlphabeticalSuggestions - Returns list of words in thesaurus that are alphabetically close.

GetAllSynonyms - Returns list of all synonyms (words with same meaning).

GetMatchingTerms - Returns list of Term objects each with their own synonyms and antonyms.

Examples

Below are some use scenarios of the thesaurus components supplied, please refer to the API documentation for additional detail.

Simple Usage, Dialog Thesaurus

To simply activate the Thesaurus dialog through a menu item;

1. Drag the Thesaurus control onto the form.

2. Set it's TextBoxBase property to the TextBox that the thesaurus should work with.

3. Add a menu or button, that the user can activate the Thesaurus from.

4. Launch the Thesaurus from the menu or button click handler:

```
thesaurus1.ShowDialog();
```

Simple Usage, Adding A ContextMenuStrip (.NET2+)

The Thesaurus component has a method called "GetSuggestionToolStripItems" which returns a list of ToolStripItem objects for either synonyms/antonyms or related words (if synonyms aren't found). These menu items can easily be added to a sub context menu (for example) in the popup handler of the text box's context menu strip;

[From the point left off in the "Dialog Thesaurus" example above.]

1. Add a context menu strip and assign it to the text box.

2. Add a menu item to the context menu with text "Synonyms" or "Thesaurus", this will be where the submenu is attached, call it "synonymsToolStripMenuItem".

3. Create an event handler for the context menu's Opening event, which will request the thesaurus suggestions and add them as a submenu;

C#

```
    private void contextMenuStrip1_Opening(object sender, CancelEventArgs e)
      {
          ToolStripItem[] items =
this.thesaurus1.GetSuggestionToolStripItems();
        this.synonymsToolStripMenuItem.DropDownItems.Clear();
        if (items == null)
        {
          ToolStripMenuItem noSuggs = new ToolStripMenuItem("No
suggestions");
           noSuggs.Enabled = false;
          synonymsToolStripMenuItem.DropDownItems.Add(noSuggs);
        }
        else
        {
          synonymsToolStripMenuItem.DropDownItems.AddRange(items);
           synonymsToolStripMenuItem.DropDownItems.Add(new
ToolStripSeparator());
          synonymsToolStripMenuItem.DropDownItems.Add("&Thesaurus...",
null, new EventHandler(thesaurusItem_Click));
        }
      }
```

VB.NET

```
Private Sub contextMenuStrip1_Opening(ByVal sender As Object, ByVal e As
CancelEventArgs) _

Handles contextMenuStrip1.Opening

Dim items() As ToolStripItem = Me.thesaurus1.GetSuggestionToolStripItems

Me.synonymsToolStripMenuItem.DropDownItems.Clear

If (items Is Nothing) Then

Dim noSuggs As ToolStripMenuItem = New ToolStripMenuItem("No suggestions")

noSuggs.Enabled = false

synonymsToolStripMenuItem.DropDownItems.Add(noSuggs)

Else

synonymsToolStripMenuItem.DropDownItems.AddRange(items)
```



```
synonymsToolStripMenuItem.DropDownItems.Add(New ToolStripSeparator)

synonymsToolStripMenuItem.DropDownItems.Add("&Thesaurus...", Nothing,
AddressOf Me.thesaurusItem_Click)

End If

End Sub
```

4. Add the thesaurusItem_Click handler

C#

```
void thesaurusItem_Click(object sender, EventArgs e)
 {
    this.thesaurus1.DialogForm.TopMost = true;
    this.thesaurus1.ShowDialog();
 }
```

VB.NET

```
Private Sub thesaurusItem_Click(ByVal sender As Object, ByVal e As
EventArgs) Handles thesaurusItem.Click

Me.thesaurus1.DialogForm.TopMost = true

Me.thesaurus1.ShowDialog

End Sub
```

Simple Usage, Adding A Context Menu (.NET1+)

The Thesaurus component has a method called "GetSuggestionMenuItems" which returns a list of MenuItem objects for either synonyms/antonyms or related words (if synonyms aren't found). These menu items can

easily be added to a sub context menu (for example) in the popup handler of the text box's context menu;

[From the point left off in the "Dialog Thesaurus" example above.]

1. Add a context menu and assign it to the text box.

2. Add a menu item to the context menu with text "Synonyms" or "Thesaurus", this will be where the sub-menu is attached, call it "synonymsMenuItem".

3. Create an event handler for the context menu's popup event.

C#

```
contextMenu1.Popup += new System.EventHandler(this.contextMenu1_Popup);

private void contextMenu1_Popup(object sender, System.EventArgs e)

{.....}
```

VB.NET

```
Private Sub contextMenu1_Popup(ByVal sender As Object, ByVal e As
System.EventArgs) Handles contextMenu1.Popup

....

End Sub
```

4. In the event handler, retrieve the synonyms from the thesaurus

C#

```
MenuItem[] suggestions = this.thesaurus1.GetSuggestionMenuItems();
```

VB.NET

```
Dim suggestions As MenuItem() = Me.thesaurus1.GetSuggestionMenuItems()
```

5. Set up an inactive "No suggestions" menu item and a "Thesaurus..." menu item to launch the dialog

C#

```
MenuItem noSuggestionsMenuItem = new MenuItem("(No suggestions)");

MenuItem thesaurusMenuItem = new MenuItem("&Thesaurus...", new
```



```
EventHandler(this.thesaurusMenuItem_Click));

noSuggestionsMenuItem.Enabled=false;

synonymsMenuItem.MenuItems.Clear();
```

VB.NET

```
Dim noSuggestionsMenuItem As New MenuItem("(No suggestions)")

Dim thesaurusMenuItem As New MenuItem("&Thesaurus...", AddressOf
thesaurusMenuItem_Click)

noSuggestionsMenuItem.Enabled = False

synonymsMenuItem.MenuItems.Clear()
```

6. Add the synonyms or "No suggestions" and "Thesaurus..." menu items to the synonymsMenuItem

C#

```
if(suggestions!=null)

  synonymsMenuItem.MenuItems.AddRange(suggestions);

else

 synonymsMenuItem.MenuItems.Add(noSuggestionsMenuItem);

synonymsMenuItem.MenuItems.Add("-");

synonymsMenuItem.MenuItems.Add(thesaurusMenuItem);
```

VB.NET

```
If Not (suggestions Is Nothing) Then

 synonymsMenuItem.MenuItems.AddRange(suggestions)

Else

  synonymsMenuItem.MenuItems.Add(noSuggestionsMenuItem)

End If
```



```
synonymsMenuItem.MenuItems.Add("-")

synonymsMenuItem.MenuItems.Add(thesaurusMenuItem)
```

7. Create an event handler for the "Thesaurus..." menu item

C#

```
void thesaurusMenuItem_Click(object sender, System.EventArgs e)

{

 this.thesaurus1.Show(thesaurus1.FocusCharacterPosition);

}
```

VB.NET

```
Sub thesaurusMenuItem_Click(ByVal sender As Object, ByVal e As
System.EventArgs)

 thesaurus1.Show(thesaurus1.FocusCharacterPosition)

End Sub
```

That's it, the complete code listing can be viewed in the Visual Studio examples that accompany the product.

Customizing The User Interface

The ThesaurusControl source is included with the product demos, therefore it can be customized in the Visual Studio designer and activated via a simple 'provider' pattern.

[From the point left off in the "Adding A Context Menu" example above.]

1. Add the ThesaurusControl source to the project and create a new Windows Form.

2. Add the ThesaurusControl to the Form and position/resize as appropriate.

3. The Thesaurus component creates new instances of the form and control through the DialogFormProvider property, which can be set to any instance of IDialogFormProvider implementing classes.

Create a new class called CustomThesaurusDialogProvider.

C#

```
class CustomThesaurusDialogProvider:
Keyoti.Thesaurus.Windows.IDialogFormProvider{

  public Keyoti.Thesaurus.Windows.ThesaurusDialog GetDialogForm(){

        return new ThesaurusDialogForm();

  }

}
```

VB.NET

```
Class CustomThesaurusDialogProvider

      Implements Keyoti.Thesaurus.Windows.IDialogFormProvider

  Public Function GetDialogForm() As Keyoti.Thesaurus.Windows.ThesaurusDialog
Implements Keyoti.Thesaurus.Windows.IDialogFormProvider.GetDialogForm

     Return New ThesaurusDialogForm

  End Function 'GetDialogForm

End Class
```

4. Set the DialogFormProvider property (eg. in the Form constructor).

C#

```
thesaurus1.DialogFormProvider = new CustomThesaurusDialogProvider();
```

VB.NET

```
thesaurus1.DialogFormProvider = New CustomThesaurusDialogProvider
```

5. Whenever the Thesaurus.Show() or Thesaurus.ShowDialog() methods are called, the custom form and control will be used.

Third Party Text Controls

The Keyoti.Thesaurus.Windows.IThesaurusableTextComponent interface defines requirements for text boxes to work with the thesaurus. Typically it is not possible for developers to implement interfaces directly in the text box class, however simple 'adapter' classes can be written to map the interface requirements to the text box. The projects include example uses with TXTextControl via the ThesaurusTXAdapter control.

Requests for development of and assistance with writing adapter classes should be sent to support@keyoti.com.

Conclusion

Thank you for using the Thesaurus component, we hope that you experience much success with it. Keyoti is committed to improving all of it's products and truly values customer feedback of any kind. If you would like to contact us about any of our products please do so through our web site.

http://www.keyoti.com/contact.html

We thrive on suggestions for components and feature requests, if you have a component wish please don't hesitate to get in touch, it may be the next thing we develop!