Logic for tbxInput\_KeyU

Preconditions:

1. Adding
2. Editing
3. Searching

External Variables:

1. bool LongKeysOK {Key words can be longer that 50 characters}
2. bool AddCommentToReusedKeyWord
3. string ReusedKeyWord
4. bool LongKeyWordsOK

Internal Variables:

1. bool KeyWord
2. bool Comment
3. string CurrentKeyWord

New Private Methods:

1. ShowKeyWordsBeginningWithThisString();
2. ShowSearchMessage() ;
3. bool CommentIsValid(string InputText);
4. AddThisCommentToTheReusedKeyWord();
5. AddNewKeyWordToKeyWordList();

Validity Checks:

1. If KeyWord:

a. LenGT50 {Is Length>50}. If set to false then when the length is greater that 50 a Message

{Show Length Message} appears asking the user to set the LenGT50 variable.

b. If rbtSearch.IsChecked && lbxKeyWords.Text = “” ShowSearchMessage.

2. If Comment

a. If contains ‘;’ && contains ‘#’ Show HashAndSemicolon Message

b. If contains ‘;’ Show Contains Semicolon Message

c. If contains ‘#’ Show HashMessage

Program flow logic

private static bool AddCommentToReusedKeyWord = false; //If true the comment should be added to the ReusedKeyWord

private static bool LongKeyWordsOK = false;// if true Key words can be longer that 50 characters

private string ReusedKeyWord = "";//This keyword has beed entered more that once

private void tbxInput\_KeyUp(object sender, KeyEventArgs e)

{

//Receives Key e

// Code to clear tbxInput if backspace results in empty text

if (e.Key == Key.Back)

{

if (tbxInput.Text == "") lbxKeyWords.Items.Clear();

}

//Remove any spaces at the end of the input text

tbxInput.Text. Trim();

//Clear all the items in the list of key words

lbxKeyWords.Items.Clear();

if(char 0 == ‘#’) Comment = true;

else KeyWord = true;

//Process KeyWord

if(KeyWord)

{

//If key != Enter

if( e.Key != Enter)

{

//Shows all the key words in the dictionary that begin with the string in the lbxKeyWords

ShowKeyWordsBeginningWithThisString();

if(tbxInputText.Text == "" return;

//If searching check to see if there are no items in the lbxKeyWords

if(rbtSearch.IsChecked)

{

if(lbxKeyWords.Items.Count == 0);

//Show message and return

ShowSearchKeyWordInvalid();

return;

}//End if(rbtSearch.IsChecked

}

else

{ // key word entered

CurrentKeyWord = tbxInput.Text;

//Check to see if the user wants to enter a key word > 50 chars

if((LengthOfKeyWordGrThan50) && (!LongKeyWordsOK))

{

SetLongKeyWordBool();

if(!LongKeyWordsOK) return;

} //End

//Check to see if this key word has already been used

if(CurrentKeyWord == ReusedKeyWord)

{

ReusedKeyWord = CurrentKeyWord;

AddCommentToReusedKeyWord = true;

//Clear Input textbox and lbxKeyWords.Items

tbxInput.Text = “”;

lbxKeyWords.Items.Clear;

return;

} //End if(CurrentKeyWord == ReusedKeyWord)

else

{ //This is a keyword

//Check to see if this keyword is NOT in KeyWordsStaticMembers.KeyWordList

if(!KeyWordsStaticMembers.KeyWordList.Contains(CurrentKeyWord))

AddNewKeyWordToKeyWordList(CurrentKeyWord);

//Add the current keyword to tbxAllKeyWords

tbxAllKeyWords.Text = tbxAllKeyWords.Text + CurrentKeyWord + ‘;’;

//Clear Input textbox and lbxKeyWords.Items

tbxInput.Text = “”;

lbxKeyWords.Items.Clear;

return;

} //End else This is a keyword

} //End key word endered

} //End if KeyWord

//Process Comments

if(Comment)

{

//Determine whether the comment contains a ‘;’ or a ‘#’ and if so will return the used to the main screen

if(!CommentIsValid(tbxInput.Text)) return;

if(e.Key == Enter) //The comment is complete

{ // Entering a new Comment

/\* Determine if this comment should be added to a previously used key word

if so, then call AddThisCommentToTheReusedKeyWord and clear the varialbes\*/

if(AddCommentToReusedKeyWord)

{

AddThisCommentToTheReusedKeyWord((tbxInput.Text)

tbxInput.Text = “”;

lbxKeyWords.Items.Clear;

return;

} //End add comment to rused key word

} //End if(AddCommentToReusedKeyWord

else

{ //Add this comment to tbxAllKeyWords.Text

tbxAllKeyWords.Text = tbxAllKeyWords.Text + tbxInput.Text + ';';

tbxInput.Text = “”;

lbxKeyWords.Items.Clear;

return;

} //End Add this comment to tbxAllKeyWords.Text

} //End If comment

} //End tbxInput\_KeyUp

//= = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

#region private methods for tbxInput\_KeyUp

private static SetLongKeyWordBool()

{

MessageBox.Show("Your KeyWord is >50 Charcters. Do you want to store it?", "No", MessageBoxButton.YesNo);

MessageBoxResult result = MessageBox.Show("Your KeyWord is > 50 Charcters.Do you want to store it ? ", "No", MessageBoxButton.YesNo);

switch (result)

{

case MessageBoxResult.Yes:

LongKeyWordsOK = true;

case MessageBoxResult.No:

LongKeyWordsOK = false;

}

return

}

private static ShowSearchKeyWordInvalid()

{

// If lbxKeyWords is empty in the Search statge send a message that you can only accepts existing keywords

if (lbxKeyWords.Items.Count == 0)

{

MessageBox.Show("When You are in the Search mode you can only search for existing Keywords");

return;

}

} //End ShowSearchKeyWordInvalid

private static ShowKeyWordsBeginningWithThisString()

{

string KeyWord = "";

/\* Determine if there are Keywords showing in lbxKeyWords then select #0,

\* Set LinkNoteStaticMembers.SelectedKeyWord to this value,

\* Add the keyword to the tbxAllKeyWords.Text,

\* and return \*/

if (lbxKeyWords.Items.Count != 0)

//There is data in the lbxKeyWords ListBox

{

// Create a list from the KeyWords in lbxKeyWords so that the 0th item can be chosen

List<string> myCurrentKeyWordsList = new List<string>();

// Populate myCurrentKeyWordsList with the selected columns.

foreach (string thisKeyWordItem in lbxKeyWords.Items)

{

myCurrentKeyWordsList.Add(thisKeyWordItem);

}

//Set the Selected KeyWord to the 0th entry

KeyWord = myCurrentKeyWordsList[0];

//Set the LinkNoteStaticMembers.SelectedKeyWord string to this value

LinkNoteStaticMembers.SelectedKeyWord = KeyWord;

LinkNoteStaticMembers.SearchKeyWord = KeyWord;

// add this Keyword to the list of selected keywords

if (LinkNoteStaticMembers.KeyWordSearch)

tbxAllKeyWords.Text = tbxAllKeyWords.Text + KeyWord + ';';

//Clear tbxInput

tbxInput.Text = "";

//Clear lbxKeyWords

lbxKeyWords.Items.Clear();

return;

}// End there is data in the lbxKeyWords ListBox

} //End ShowKeyWordsBeginningWithThisString()

private static bool CommentIsValid (string InputText)

{

bool Valid = true;

//Test to insure that the text does not contain a semicolon ;

if (InputText.Contains(";"))

{

MessageBox.Show("You cannot include a semicolon ';' in a key word or comment!");

Valid = false;

return Valid;

}

//Test to see if text contains "#"

if (InputText.Contains("#"))

{

//This may be a comment

int numberOfHashes = StringHelper.ReturnNumberOfDeliniters(text, '#');

//If there is only one and its position is 0 this is a valid comment

if (numberOfHashes == 1)

{

if (text.StartsWith("#"))

{

return Valid;

}

else

{

MessageBox.Show("The Hash mark # must start a comment!");

Valid = false;

return Valid;

}

}

else if(numberOfHashes > 1)

{

MessageBox.Show("You can only have 1 Hash mark in an expression!");

Valid = false;

return Valid;

}

}// end text contains #

} // CommentIsValid ()

private static AddThisCommentToTheReusedKeyWord(string NewCommentToAdd)

{

//Use Split to convert the text in tbxAllKeyWords.Text into a string array

string[] ArrayOfAllKeyWordsItems = tbxAllKeyWords.Split(';');

//Create a Dictionary where the Keys are the KeyWords and the value are the Comments attached to that key word

Dictionary<string,string> KeyWord\_CommentsDictionary = new Dictionary<string,string>();

string currentKeyWord = "";

foreach(string Item in ArrayOfAllKeyWordsItems)

{

if(Iten.IndexOf('#') == -1)

{

//This is a KeWord so creat a new Dictionary item

KeyWord\_CommentsDictionary.Add(Item,"");

currentKeyWord = Item;

} //End of if(Iten.IndexOf('#') == -1)

if(Iten.IndexOf('#') == 0)

{

//This is a comment so add it to the Dictionary item whose key = currentKeyWord

string CurrentValue = KeyWord\_CommentsDictionary[currentKeyWord];

CurrentValue = CurrentValue+Item+';';

//Add the updated value to the dictionary

KeyWord\_CommentsDictionary[currentKeyWord] = CurrentValue;

}//End if(Iten.IndexOf('#') == 0)

}//End of foreach(string Item in ArrayOfAllKeyWordsItems

string CurrentValueOfReusedKeyWord = KeyWord\_CommentsDictionary[ReusedKeyWord];

CurrentValueOfReusedKeyWord =CurrentValueOfReusedKeyWord + NewCommentToAdd +';';

KeyWord\_CommentsDictionary[ReusedKeyWord] =CurrentValueOfReusedKeyWord;

//Change the Dictionary back into a ';' delimited string

string RevisedtbxAllKeyWordsText = "";

//cycle thru the Dictionary getting all items

foreach(KeyValuePair KVP in KeyWord\_CommentsDictionary)

{

string Key = KVP.Key;

string Value - KVP.Value;

RevisedtbxAllKeyWordsText = RevisedtbxAllKeyWordsText+Key+';'+value+';';

}//End foreach(KeyValuePair KVP in KeyWord\_CommentsDictionary

tbxAllKeyWords.Text = RevisedtbxAllKeyWordsText;

return;

} // AddThisCommentToTheReusedKeyWord()

private static AddNewKeyWordToKeyWordList(string CurrentKeyWord);

{

CurrentKeyWord.Trim();

// add this new key word to the KeyWordList

CurrentKeyWord.Trim();

// add this new key word to the KeyWordList

KeyWordsStaticMembers.KeyWordList.Add(CurrentKeyWord.Text);

// Append this new Keyword to the Keyword txt Fild

KeyWordsStaticMembers.AppendNewKeyWord(CurrentKeyWord);

//Add this new keyword to the SortedListOfKeyWords.txt file

KeyWordsStaticMembers.AppendNewSortedKeyWord(CurrentKeyWord);

// Convert Keyword to Dictionary Item by replacing all spaces with '\_'

string thisKeyWord = CurrentKeyWord;

//Convert the keyword to the dictionary key format (no spaces)

string ConvertedThisKeyWord = thisKeyWord.Replace(' ', '\_');

//Add the new converted Key word to the dictionary with a value containing only the starting delimiter, ;

KeyWordsStaticMembers.KeyWordsDictionary.Add(ConvertedThisKeyWord, ";");

// Add the new converted keyword to the NotesDictionary.txt file

KeyWordsStaticMembers.AppendNewKeyWordDictionaryItemString

} // End AddNewKeyWordToKeyWordList();

#endregion private methods for tbxInput\_KeyUp