Документация: Система за подсказки

Съдържание

1. Увод
2. Проектиране и реализация
3. Потребителско ръководство
4. Приложение

* 1. Увод

Подсказките при попълване на текст са широко разпространени във всякакви приложения. Те улесняват потребителите и спестяват значително количество време. Изграждането на добра система за подсказки е предизвикателство пред разработчиците на приложения и не трябва да бъде пренебрегвано.

* 1. Проектиране и реализация

Системата за подсказки се състои от два основни класа: SearchSuggestions и EnterSuggestions.

SearchSuggestions представлява набор от статични методи, предназначени за използване тогава, когато се търси нещо по дадена само част от него и резултатите могат да бъдат много. Ако разработчик желае да използва този клас, той трябва да направи следното:

* + Да създаде контрола, която да визуализира подсказките
  + Да свърже към нея property, в което да се съхраняват подсказките

<ListBox ... ItemsSource="{Binding Source={StaticResource vm}, Path=Suggestions}" ...

></ListBox>

* + Да свърже Text property-то на контролите за въвеждане на данни с такова в ViewModel-а

<TextBox ... Text="{Binding Source={StaticResource vm}, Path=CatNameTextBox,

UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" ... ></TextBox>

# • В set частта на това property да извика SwitchContext() и Suggest()

public string CatNameTextBox { get { return \_catNameTextBox; } set { \_catNameTextBox = value;

SearchSuggestions.SwitchContext(this, "cat\_names.txt",

"CatNameTextBox", "CatNameIsFocused", CatsList, Suggestions);

SearchSuggestions.Suggest(this);

PropChanged("CatNameTextBox");

}

}

• Параметрите в SwitchContext() представляват:

# 1. Текущият ViewModel

1. Име на файл за съхранение на подсказките
2. Име на property-то, което съхранява текста на контролата
3. Име на property-то, което отговаря за фокуса върху контролата (ще бъде разяснено по-нататък)
4. Име на property, което съдържа резултата от търсенето
5. Име на property-то, в което се съхраняват подсказките

* На Suggest() се подава само текущият ViewModel
* За да може при избор на подсказка, тя да се попълни в полето, трябва да създаде команда, в която да извика Select() и да я използва по следния начин:

private void SelectSuggestion()

{

SearchSuggestions.Select(this, \_selectedSuggestion); }

public RelayCommand SelectSuggestionCommand { get; set; }

SelectSuggestionCommand = new RelayCommand(SelectSuggestion);

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}">

<TextBlock.InputBindings>

<MouseBinding Command="{Binding Source={StaticResource vm}, Path=SelectSuggestionCommand}" MouseAction="LeftDoubleClick"/>

</TextBlock.InputBindings>

</TextBlock>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.InputBindings>

<KeyBinding Command="{Binding Source={StaticResource vm},

Path=SelectSuggestionCommand}" Key="Enter"/>

</ListBox.InputBindings>

• Параметрите на Select() представляват:

# 1. Текущият ViewModel

2. Property, съхраняващо избраната подсказка

EnterSuggestions изработва подсказки за цели обекти. Ако разработчик желае да използва този клас, той трябва да направи следното:

* Да създаде обект от този клас за всеки модел:

EnterSuggestions userSuggestions = new EnterSuggestions("users.txt");

* На конструктора се подава име на файл с подсказките
* Да създаде контрола, която да визуализира подсказките
* Да свърже към нея property, в което да се съхраняват подсказките

<ListBox ... ItemsSource="{Binding Source={StaticResource vm}, Path=Suggestions}" ...

></ListBox>

* Да свърже Text property-то на контролите за въвеждане на данни с такова в ViewModel-а

<TextBox ... Text="{Binding Source={StaticResource vm}, Path=UsernameTextBox,

UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}" ... ></TextBox>

# • В set частта на това property да извика SwitchContext() и Suggest()

public string UsernameTextBox

{ get {

return CurrentUser.Username;

} set

{

CurrentUser.Username = value;

enterSuggestions.SwitchContext(this, CurrentUser, "Username",

"UsernameIsFocused", Suggestions);

enterSuggestions.Suggest(CurrentUser,propertiesToHide);

PropChanged("UsernameTextBox");

}

}

• Параметрите на SwitchContext() са:

# 1. Текущият ViewModel

1. Текущият модел
2. Име на полето в текущия модел, за което се отнася контролата за въвеждане
3. Име на property-то, което отговаря за фокуса върху контролата (ще бъде разяснено по-нататък)
4. Property, което съдържа подсказките • Параметрите на Suggest() са:
5. Текущият модел
6. Списък с имена на property-та, които не трябва да се показват на потребителя

• За да може при избор на подсказка, тя да се попълни във всички полета, трябва да създаде команда, в която да извика Select() и да я използва по следния начин:

private void SelectSuggestion()

{

enterSuggestions.Select(this, CurrentUser, \_selectedSuggestion);

PropChanged("UsernameTextBox");

PropChanged("PasswordTextBox");

PropChanged("EmailTextBox");

PropChanged("CatIDTextBox");

} public RelayCommand SelectSuggestionCommand { get; set; }

SelectSuggestionCommand = new RelayCommand(SelectSuggestion);

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}">

<TextBlock.InputBindings>

<MouseBinding Command="{Binding Source={StaticResource vm}, Path=SelectSuggestionCommand}" MouseAction="LeftDoubleClick"/>

</TextBlock.InputBindings>

</TextBlock>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.InputBindings>

<KeyBinding Command="{Binding Source={StaticResource vm},

Path=SelectSuggestionCommand}" Key="Enter"/>

</ListBox.InputBindings>

• Параметрите на Select() са:

# 1. Текущият ViewModel

1. Текущият модел
2. Property, съхраняващо избраната подсказка

FocusHelper е клас, който подпомага предаването на фокуса между текстовите полета и контролите с подсказки. Ако разработчик желае да използва този клас, той трябва да направи следното:

• Да създаде attached property, което връща true или false в зависимост от това дали контролата, с която е свързано, е фокусирана. Примерна реализация и свързване:

public static class FocusExtension

{

public static bool GetIsFocused(DependencyObject obj)

{

return (bool)obj.GetValue(IsFocusedProperty);

}

public static void SetIsFocused(DependencyObject obj, bool value)

{

obj.SetValue(IsFocusedProperty, value);

}

public static readonly DependencyProperty IsFocusedProperty =

DependencyProperty.RegisterAttached(

"IsFocused", typeof(bool), typeof(FocusExtension), new UIPropertyMetadata(false, OnIsFocusedPropertyChanged)); private static void OnIsFocusedPropertyChanged(DependencyObject d,

DependencyPropertyChangedEventArgs e)

{

var uie = (UIElement)d; if (!((bool)e.NewValue)) return; uie.Focus(); Keyboard.Focus(uie);

uie.LostFocus += UieOnLostFocus;

}

private static void UieOnLostFocus(object sender, RoutedEventArgs routedEventArgs) {

var uie = sender as UIElement; if (uie == null) return;

uie.LostFocus -= UieOnLostFocus; uie.SetValue(IsFocusedProperty, false); }

}

<TextBox ... local:FocusExtension.IsFocused="{Binding Source={StaticResource vm},

Path=UsernameIsFocused, Mode=TwoWay}" ... ></TextBox>

* Да създаде команда, в която да използва SetFocus():

private void FocusSuggestionsBox()

{

FocusHelper.SetFocus(this, "SuggestionsIsFocused");

}

public RelayCommand FocusSuggestionsBoxCommand { get; set; }

FocusSuggestionsBoxCommand = new RelayCommand(FocusSuggestionsBox);

* Да извиква командата при събитие по негова преценка. Примерна реализация:

<TextBox.InputBindings>

<KeyBinding Command="{Binding Source={StaticResource vm},

Path=FocusSuggestionsBoxCommand}" Key="Down"/>

</TextBox.InputBindings>

1. Потребителско ръководство

При въвеждане в поле, потребителят вижда до 5 от последните въведени от него фрази или цели обекти. Работата с тях се определя от разработчика и се подпомага от FocusHelper класа. В примерните реализации по-горе прехвърлянето на фокуса от полето за въвеждане към контролата за подсказки става чрез натискане на стрелка „надолу“, а избор на подсказка и връщане на фокуса – чрез клавиш „Enter“ или чрез двоен клик с левия бутон на мишката. Тези събития могат да бъдат и други в зависимост от предпочитанията на разработчика.

1. Приложение

# SearchSuggestions.cs

namespace SuggestionsSystem

{ public static class SearchSuggestions

{

static string filename; static PropertyInfo InputProperty; static PropertyInfo TextBoxFocusProperty;

static ObservableCollection<string> ResultProperty { get; set; } static ObservableCollection<string> SuggestionsProperty { get; set; }

public static void TrySaveSuggestion(object targetVM)

{

if (ResultProperty.Count() > 0)

{ try

{

File.AppendAllText(@"Suggestions\" + filename,

InputProperty.GetValue(targetVM).ToString() + "\n");

}

catch (DirectoryNotFoundException)

{

Directory.CreateDirectory("Suggestions");

File.AppendAllText(@"Suggestions\" + filename,

InputProperty.GetValue(targetVM).ToString() + "\n");

}

} }

public static void SwitchContext(object targetVM, string filename, string inputPropertyName, string textBoxFocusProperty, ObservableCollection<string> resultProperty, ObservableCollection<string> suggestionsProperty)

{

SearchSuggestions.filename = filename;

InputProperty = targetVM.GetType().GetProperty(inputPropertyName); TextBoxFocusProperty =

targetVM.GetType().GetProperty(textBoxFocusProperty); ResultProperty = resultProperty;

SuggestionsProperty = suggestionsProperty;

}

public static void Suggest(object targetVM)

{

SuggestionsProperty.Clear(); try

{

IEnumerable<string> lines = File.ReadAllLines(@"Suggestions\" + filename).Reverse();

foreach (string str in lines)

{

if (str.Contains(InputProperty.GetValue(targetVM).ToString()) &&

!SuggestionsProperty.Contains(str))

{

if (SuggestionsProperty.Count<5)

{

SuggestionsProperty.Add(str);

} else { break; }

}

}

} catch (FileNotFoundException) { }

}

public static void Select(object targetVM, string value)

{

if (value != null)

{

InputProperty.SetValue(targetVM, value);

FocusHelper.SetFocus(targetVM, TextBoxFocusProperty.Name); }

}

}

}

# EnterSuggestions.cs

namespace SuggestionsSystem

{

public class EnterSuggestions

{

string filename;

PropertyInfo InputProperty;

PropertyInfo FocusProperty;

ObservableCollection<string> SuggestionsProperty { get; set; }

public EnterSuggestions(string filename)

{

this.filename = filename;

} public void SwitchContext(object targetVM, object model, string inputPropertyName, string focusPropertyName, ObservableCollection<string> suggestionsProperty)

{

InputProperty = model.GetType().GetProperty(inputPropertyName);

FocusProperty = targetVM.GetType().GetProperty(focusPropertyName);

SuggestionsProperty = suggestionsProperty;

}

public void SwitchContext(object targetVM, string inputPropertyName)

{

InputProperty = targetVM.GetType().GetProperty(inputPropertyName);

}

public void SetSuggestionsProperty(ObservableCollection<string> suggestionsProperty)

{

SuggestionsProperty = suggestionsProperty;

}

public void TrySaveSuggestion(object model, Func<string, bool> validation, string modelPropertyToValidate)

{ try {

string suggestionString = "";

PropertyInfo[] properties = model.GetType().GetProperties(); foreach (PropertyInfo property in properties)

{

if (property.Name.Equals(modelPropertyToValidate))

{

if (!validation(property.GetValue(model).ToString()))

{

MessageBox.Show("Validation failed!"); return;

} }

suggestionString = suggestionString + property.Name + ":" + property.GetValue(model) + "#";

}

File.AppendAllText(@"Suggestions\" + filename, suggestionString + "\n");

} catch (System.NullReferenceException e) { }

}

public void Suggest(object model, IEnumerable<string> propertiesNotToShow) {

SuggestionsProperty.Clear();

string currentPropertyName = InputProperty.Name;

Regex regex = new Regex(currentPropertyName + @":(.+?)#"); try

{

IEnumerable<string> lines = File.ReadAllLines(@"Suggestions\" + filename).Reverse();

foreach (string str in lines)

{

Match match = regex.Matches(str)[0];

string foundPropertyValue = match.Groups[1].ToString(); if

(foundPropertyValue.Contains(InputProperty.GetValue(model).ToString()) &&

!InputProperty.GetValue(model).ToString().Equals("") &&

!SuggestionsProperty.Contains(str))

{

if (SuggestionsProperty.Count < 5)

{

string cleanString = str;

foreach (string propertyToHide in propertiesNotToShow)

{

Regex hidePropertyRegex = new Regex(propertyToHide +

@":(.+?)#");

Match hideMatch = hidePropertyRegex.Matches(str)[0]; string hideStr = hideMatch.Groups[0].ToString(); cleanString = cleanString.Replace(hideStr, "");

}

string newStr = cleanString.Replace('#', ' ');

SuggestionsProperty.Add(newStr);

}

else { break; }

}

} }

catch (FileNotFoundException) { }

}

public void Select(object targetVM, object model, string value)

{

if (value!=null)

{

char[] separators = { ' ' };

string[] splitValue = value.Split(separators,

StringSplitOptions.RemoveEmptyEntries);

IEnumerable<string> lines = File.ReadAllLines(@"Suggestions\" + filename).Reverse();

string fullObject = null;

foreach (string objectString in lines)

{

bool isCorrectObject = true;

foreach (string partOfObject in splitValue)

{

if (!objectString.Contains(partOfObject))

{

isCorrectObject = false; break;

} }

if (isCorrectObject)

{

fullObject = objectString.Replace(" ", " #"); break;

}

}

PropertyInfo[] properties = model.GetType().GetProperties(); foreach (PropertyInfo property in properties)

{

Regex regex = new Regex(property.Name + @":(.+?)#"); Match match = regex.Matches(fullObject)[0]; string propertyValue = match.Groups[1].ToString(); try {

property.SetValue(model, propertyValue);

}

catch (ArgumentException)

{

int intVal = Int32.Parse(propertyValue); property.SetValue(model, intVal);

}

}

FocusHelper.SetFocus(targetVM, FocusProperty.Name);

}

}

}

}

# FocusHelper.cs

namespace SuggestionsSystem

{

public static class FocusHelper

{

public static void SetFocus(object targetVM, string propertyToFocus)

{

PropertyInfo focusProperty;

focusProperty = targetVM.GetType().GetProperty(propertyToFocus); focusProperty.SetValue(targetVM, true);

}

}

}