

## **Лабораторная работа 8. Повышение удобства использования приложений**

### **Цель работы**

Изучение средств для повышения удобства работы пользователей и получение навыков по созданию контекстной справки, всплывающей подсказки, а также файлов со справочной информацией.

### **Упражнение 1. Создание контекстной справки**

Важной частью любого приложения является понятная и точная документация. Снабдить ваше приложение справкой позволяет компонент **HelpProvider**.

1. Откройте Windows-приложение WinAsynchMethod.
2. Откройте форму в режиме конструктора.
3. Выберите пункт меню **View**  **ToolBox**.
4. Добавьте ЭУ **HelpProvider** на форму.
5. Выделите поле **txbA** для отображения ее свойств.
6. Для свойства **HelpString** on **helpProvider1** задайте значение **For input integer A.**
7. Постройте и запустите приложение.
8. Переместитесь по форме, используя клавишу **Tab**, до тех пор, пока поле **txbA** не окажется в фокусе.
9. Нажмите на клавишу **F1** для отображения контекстной справки для поля **txbA**.

Простые формы обычно в своем заголовке имеют кнопку с вопросительным знаком, при нажатии на которую курсор меняет свой вид на изображение с вопросом. При щелчке на выбранном элементе управления появляется его краткое описание (подсказка).

Создайте подобную функциональность на форме проекта WinAsynchMethod:

- Добавьте к имеющимся свойствам формы следующие свойства:

Свойство	Значение
MaximizeBox	False
MinimizeBox	False
HelpButton	True
FormBorderStyle	FixedDialog

2. Для полей ввода **txbA**, **txbB** и двух кнопок в свойстве **ShowHelp on helpProvider1** каждого из этих элементов установите значение **True**.

3. Текст, введенный в поле свойства **HelpString on helpProvider1**, будет появляться в качестве подсказки для конкретного элемента. Установите следующие значения этого свойства для каждого элемента:

txbA	For input integer A
txbB	For input integer B
btnRun	Sum
btnWork	Start work

4. Постройте и запустите приложение.

5. Для активации контекстной справки нажмите на кнопку “?”, расположенную в правом верхнем углу приложения.

6. Нажмите на любую кнопку, появится маленькое окошко, объясняющее, что происходит при ее нажатии.

## **Упражнение 2. Использование справочного файла**

- Откройте Windows-приложение WinAsynchMethod.
- Откройте форму в режиме конструктора.
- Выберите пункт меню **View**  **ToolBox**.
- Добавьте ЭУ **HelpProvider** на форму (если он не был добавлен ранее).
- В папке с решением создайте файл справки, например, документ Microsoft Word. Текст укажите произвольный.
- Для элемента **helpProvider1** в свойстве **HelpNamespace** укажите путь к файлу справки.
- Реализуйте возможность вызова файла справки созданием либо команды меню, либо кнопки (и команда меню и кнопка может называться, например, **help**).
- Создайте обработчик события выбора файла справки. В теле обработчика укажите следующую строку:  
`Help.ShowHelp(this, helpProvider1.HelpNamespace);`

9. Постройте и запустите приложение.
10. Выберите команду вызова справки. Проверьте, что открылся требуемый файл.

### **Упражнение 3. Добавление всплывающих подсказок**

Компонент *ToolTip* позволяет назначить элементам управления подсказки. Они появляются в окнах, когда мышь находится над элементом управления, и могут предоставлять пользователю краткие сведения о нем.

1. Откройте Windows-приложение WinAsynchMethod в режиме конструктора.

2. Выберите пункт меню **View→ToolBox**.

3. Добавьте на форму элемент управления **ToolTip**.

4. В окне **Properties** расположенных на форме элементов и в самой форме появилось свойство **ToolTip on toolTip1**. Установите следующие значения этого свойства для каждого из элементов:

txbA	For input integer A
txbB	For input integer B
btnRun	Sum
btnWork	Start work

5. Постройте и запустите приложение. Проверьте, что при наведении курсора на элемент управления появляется его подсказка.

### **Упражнение 4. Автоматический выбор языка при запуске приложения**

При распространении приложения часто бывает необходимо обеспечивать пользователям возможность работать в своей языковой среде. В связи с этим, при разработке приложений приходится задумываться о переводе пользовательского интерфейса на другие языки. На практике используется два способа решения данной проблемы, первый: создается локальная версия целиком на одном языке и второй: программы содержат многоязычный интерфейс, позволяющий менять оформление приложения непосредственно в ходе работы.

В ходе установки операционной системы Windows при определении региональных параметров пользователю предлагается выбрать язык стандартов и форматов. Выбранное значение доступно для изменения в дальнейшем — меню “Пуск” | “Панель управления” | “Язык и региональные параметры” | вкладка “Региональные параметры”.

В этом упражнении вы создадите приложение, которое автоматически будет определять установленный язык стандартов и выводить соответствующий пользовательский интерфейс.

1. Создайте новое Windows-приложение и назовите его WinLanguage.

2. Добавьте на форму кнопку и главное меню.

3. Установите следующие значения свойства **Text** для элементов:

главное меню – **menu**,

первая команда – **command one**,

вторая команда – **command two**,

кнопка – **Close**,  
для самой формы – **Form**.

4. Для кнопки реализуйте обработчик, закрывающий форму:  
`this.Close();`

5. Для формы установите свойству **Localizable** значение **True**. Это свойство разрешает поддержку многоязычного интерфейса.

6. В свойстве **Language** выберите значение **English (United States)**.

7. Постройте приложение. В итоге получилась версия программы с интерфейсом на английском языке.

8. В свойстве **Language** выберите **Russian (Russia)**.

9. Измените свойство **Text** элементов, заменив названия на английском языке соответствующими названиями на русском.

10. Перейдите в код формы и подключите пространство имен **Threading**:

```
using System.Threading;
```

11. В конструкторе формы до `InitializeComponent()`; установите культуру пользовательского интерфейса равной текущей культуре:

```
Thread.CurrentThread.CurrentCulture =  
    Thread.CurrentThread.CurrentCulture;
```

12. Постройте и запустите приложение. Среда CLR проверяет установленный язык и выводит приложение с интерфейсом на языке, установленном на вкладке "Региональные параметры" в настройках инструмента "Язык и региональные параметры".

13. Закройте приложение. Перейдите в "Панель управления" и установите другой язык (например, "Английский (США)") на вкладке "Региональные параметры" в настройках инструмента "Язык и региональные параметры". Снова запустите приложение. Теперь интерфейс приложения должен быть на другом языке (например, английском).

14. При локализации приложения среда Visual Studio .NET создает сборку, в которой хранятся все данные о приложении. В окне Solution Explorer нажмите на кнопку (**Show All Files**) для просмотра добавленных файлов. Названия файлов-ресурсов (`Form1.en-US` и `Form1.ru-RU`) содержат в себе указание на язык (первая часть – en или ru) и регион (вторая часть US или RU).

### **Упражнение 5. Локализация приложения**

Реализовать локализацию, т.е. предоставить пользовательский интерфейс, характерный для текущего региона, можно с помощью встроенных в Visual Studio средств локализации.

Visual Studio позволяет создавать альтернативные версии культурозависимых форм и автоматически управляет поиском ресурсов, соответствующих данной культуре.

Для пользовательского интерфейса культура предоставляется экземпляром `CultureInfo` и отличается от свойства

*CulturInfo.CurrentCulture*. В то время как оно определяет формат, применяемый к системно-форматируемым данным, *CurrentUICulture* определяет ресурсы, загружаемые в локализованные формы во время выполнения. Культура пользовательского интерфейса устанавливается в свойстве *CurrentThread*. *CurrentUICulture*

В этом упражнении вы локализуете пользовательский интерфейс приложения и добавите в него локализованные строковые ресурсы.

### Локализация формы Windows-приложения

1. Откройте стартовый проект UsabilityDemo.sln из папки **install\_folder\Practices\Mod08\Mod08\_04\Starter\_2008**.

2. Откройте файл UsabilityDemo.cs в режиме конструктора.

3. Нажмите на кнопку **Show All Files** в окне **Solution Explorer**.

4. Обратите внимание на значение свойства формы **Localizable**. Оно установлено в **True**, что означает, что форма может быть локализована.

5. Для свойства формы **Language** задайте значение **French(France)**.

В данный момент вы видите английскую версию формы, но теперь вы можете преобразовать ее во французскую. Обратите внимание на то, что в окне **Solution Explorer** под файлом UsabilityDemo.cs появилось несколько новых файлов ресурсов (UsabilityDemo.fr.resx и UsabilityDemo.fr-FR.resx). Обратите внимание на то, что данная форма также была локализована для Германии и Японии.

6. Для свойства формы **Text** задайте значение **Démonstration de l'utilisation**.

**Замечание:** Текст, необходимый для создания французской версии формы можно скопировать из файла UsabilityDemoLocalizedStrings.rtf, расположенного в папке **install\_folder\Practices\Mod08\Mod08\_04**.

7. Для свойства **Text** меню **Help** и пункта меню **Help** задайте значение **Aide**.

8. Для свойства **Text** кнопки **Choose a Culture** задайте значение **Choisir une langue**.

9. Для свойства **Text** кнопки **Show Date/Time** задайте значение **Afficher la date/l'heure**.

10. Для свойства **Text** кнопки **Show Currency** задайте значение **Afficher la devise**.

11. Для свойства **Text** кнопки **Show a String** задайте значение **Afficher une chaîne**.

12. Для свойства **Text** кнопки **Exit** задайте значение **Quitter**.

### Добавление в приложение файл строковых ресурсов

1. В окне **Solution Explorer** ПКМ по проекту **UsabilityDemo** | **Add** | **Add New Item**.

2. В окне **Add New Item** выберите **Assembly Resource File**.

3. Задайте для файла ресурсов имя **UsabilityDemoText.fr-FR.resx** и нажмите на кнопку **Open**.

4. В строке ресурсов задайте для **Name** значение **SimpleTextString**.

5. В строке ресурсов задайте для **Value** значение **Voici du texte.**
6. Сохраните и закройте файл ресурсов.

#### **Добавление кода для получения значений строковых ресурсов**

1. Откройте файл с кодом UsabilityDemo.cs.
2. В окне Task List отобразите комментарии TODO. Для этого выберите пункт меню **View | Show Tasks | All.**
3. Добавьте три директивы **using**, для возможности поддержки локализации приложения.

```
using System.Globalization;
using System.Resources;
using System.Threading;
```

4. Найдите в коде второй комментарий TODO. Объявите **private** переменную типа **ResourceManager**.

```
private ResourceManager RM;
```

5. Найдите в коде следующий комментарий TODO. Создайте экземпляр класса **ResourceManager**. Код добавляется в первый конструктор приложения:

```
RM = new ResourceManager("UsabilityDemo.UsabilityDemoText",
Assembly.GetExecutingAssembly());
```

6. Найдите в коде следующий комментарий TODO. Создайте экземпляр класса **ResourceManager**. Код добавляется во второй конструктор приложения:

```
RM = new ResourceManager("UsabilityDemo.UsabilityDemoText",
Assembly.GetExecutingAssembly());
```

7. Найдите в коде следующий комментарий TODO. Добавьте код использования менеджера ресурсов для получения строки текста из файла ресурсов и отображения ее в текстовом поле.

```
OutputTextBox.Text = RM.GetString("SimpleTextString");
```

8. Найдите в коде следующий комментарий TODO. Добавьте код для задания свойствам **Culture** и **UICulture** значений, выбранных пользователем.

```
Thread.CurrentThread.CurrentCulture = new
CultureInfo(ChosenCulture, false);
Thread.CurrentThread.CurrentUICulture = new
CultureInfo(ChosenCulture, false);
```

9. Сохраните проект.

#### **Тестирование работы приложения**

1. Постройте и запустите приложение.
2. Обратите внимание на то, что при старте приложения все, что вы можете – это либо выбрать язык, либо завершить работу приложения.
3. Нажмите на кнопку **Choose a Culture**.
4. Выберите один из языков. Если вы не выберите ни один из них, по умолчанию будет использоваться English.
5. После нажатия на кнопку **OK** на форме **Culture Chooser** эта форма исчезает и появляется та версия формы **UsabilityDemo**, которая соответствует выбранным настройкам.