В. О. Розова

ПРИЛОЖЕНИЯ ТИПА WINDOWS FORMS

Поэтапное изучение создания приложений типа Windows Forms, а также возможных для использования библиотек. Разбор добавление кода на форму. Типы System.EventArgs и System.EventHandler

Введение

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

I. Создание приложений Windows Forms

Помимо использования стандартных стилей, вы всегда можете воспользоваться сторонними библиотеками, которые позволят быстрее создавать еще более красивые дизайны для приложений.

- Специализированная библиотека Bunify
- Фреймворк WPF
- Xamarin Forms

В основе своей, создание дизайна разбивается на несколько этапов:

- Добавление объектов на главное окно;
- Добавление стилей для объектов. Можно добавить стили не только стандартные, но и стили из различных библиотек;
- Добавление обработчиков событий.

Одним из доступных способов создания приложений типа Windows Forms, является создание проектов через интегрированную среду разработки – Visual Studio. В этой мастерской существует специальный раздел, который так и называется — Windows Forms Application — и предоставляет все необходимые средства для построения приложения, причем довольно быстро, т. к. не надо пользоваться объявлениями классов, чтобы поместить тот или иной тип в форму приложения. Ее просто перетаскивают мышью в форму и располагают в нужном месте.

У приложения Windows Forms имеется своя точка входа в программу, которая не совпадает с точкой входа (метод Main()) консольного приложения. У нее свое имя. Если у консольного приложения главное окно — консольное окно, то главное окно приложения Windows Forms вы сами определяете в виде объекта класса (которому вы даете свое имя), наследуемого членом класса Form. В среде Windows Forms Application главное окно строится автоматически самой средой как наследник класса Form с именем Form1. И на экране появляется изображение того, что принято называть формой: изображение квадрата с заголовком и кнопками обычного Windows-окна. То есть это окно будущего приложения, которое называется формой. Так договорились разработчики. Если же вы к приложению добавляете другую форму (можно добавлять сколько угодно), то среда назовет ее Form2 и будет считать унаследованной от класса Form. И т. д. Понятно, почему такие имена: Form1, Form2, ... Все это делает программа, а в ней надо задавать что-то вполне определенное, поддающееся алгоритмированию. Имя плюс порядковый номер подключаемого класса — вот и имя наследника Form. Просто. Вы же можете давать главному окну и всем последующим окнам, если потребуется, свои имена. Но в примере мы оставили правило среды Windows Forms Application: главное окно-форма — это Form1, которая наследуется от класса Form. Если придется добавить новые формы, они станут наследниками класса Form с именами Form1, Form2, ...

В программе создан свой конструктор класса, параметрами которого являются имя главного окна и его размеры, задаваемые шириной и высотой окна. Кроме этого использован метод CenterToScreen(), помещающий окно в центр экрана (оно все равно будет в рамках главного окна консольного приложения, потому что главное приложение — это все-таки консольное приложение).

Для запуска приложения Windows Forms используется метод Run() из другого класса пространства имен System.Windows — из класса Application.

В частности, метод Run() в качестве аргумента имеет главное окно приложения, которое он открывает, когда запускает приложение на выполнение.

Пользовательский интерфейс создается добавлением в форму управляющих элементов: кнопок, меню, меток и т. п. Этот шаг предполагает выполнение следующих действий: 1. В классе, порожденном от Form, определяется переменная-член нужного бу-

дущего элемента интерфейса. 2. Настраиваются поведение и внешний вид элемента с помощью придания его свойствам необходимых значений. 3. Полученный элемент добавляется в контейнер ControlCollection данной формы с помощью метода Controls.Add(). Control Collection по иерархии имеет вид System.Windows.Forms.Control.ControlCollection, т. е. находится в пространстве имен System.Windows.Forms и представляет собой класс, определяющий набор (коллекцию) управляющих элементов для создания интерфейса пользователя в приложении. То есть в коллекцию этого типа добавляются создаваемые элементы пользовательского интерфейса вашего приложения. Это как бы рабочий контейнер, связанный с формой, для хранения управляющих элементов: визуально мы видим в итоге, что управляющие элементы помещены, якобы, в форму, а на самом деле физически они хранятся в контейнере ControlCollection.

Если мы работаем сразу в среде Windows Forms Application, то эта среда визуально предоставляет вам список необходимых элементов. Вам остается только щелкнуть мышью на нужном элементе и перетащить его в форму. А затем настраивать элемент, задавая (или выбирая) его свойства из списка свойств элемента, который тоже открывается в определенном окне.

II. Типы System.EventArgs и System.EventHandler

System.EventHandler — один из типов делегатов, применяемых в Windows Forms во время обработки событий. Он может лишь вызывать методы, у которых только два параметра и не каких-либо, а именно таких, что первый параметр имеет обязательно тип System.Object. Это ссылка на объект, который организовал (сгенерировал) обрабатываемое событие, а второй — обязательно должен являться ссылкой на объект типа System.EventArgs, содержащий параметры для обработки сгенерированного события.

Класс EventArgs сам по себе ничего не дает для обработки события, однако он является предком многих классов, предназначенных для обработки событий, таких, например, как: MouseEventArgs, который к предыдущему классу добавляет информацию о текущем состоянии мыши; KeyEventArgs, содержащий информацию о состоянии клавиатуры; PaintEventArgs, дополняющий EventArgs информацией о графических данных.

Розова Виктория Олеговна, студетка первого курса факультета информационной безопасности, vika290704@gmail.com.

Научный руководитель: Кукин Дмитрий Петрович, заведующий кафедрой вычислительных методов и программирования БГУИР, кандидат технических наук, доцент, kukin@bsuir.by.