Самописные визуальные компоненты

Прочитать статью на сайте

Для чего это?

В процессе каждодневной работы приходится создавать кучу новых компонентов, в широком смысле слова. Каждый UserControl формально является новым компонентом. Я создаю определенную конфигурацию компонентов из стандартного набора и использую это в программе. Я не считаю получившиеся компоненты настоящими «новыми» компонентами, потому что они полезны только для меня и только в определенной программе. Как правило, я не переношу такие компоненты в другие программы и не использую их повторно за пределами программы, где они были созданы. Но, время от времени, я обнаруживаю, что некоторые вышли удачными для многоразового использования.

Иногда я нахожу удачные примеры компонентов в других системах или в платных наборах. Если кто-то что-то сделал, другой сможет это повторить. =) Ну и ко всему прочему это отличная зарядка для мозга.

Когда и изучил более плотно смарт тэги, я решил, что некоторые компоненты могут быть улучшены. Это примерно так же как использовать Extended методы для базовых классов или создания DSL – вобщем чтобы писать код быстрее и легче. То же самое применимо и к компонентам.

Итак, вот некоторые причины по которым появляются самописные компоненты:

- Случайно – как результат проектирования интерфейса в целом;
- Выделение общих частей из случайных компонентов;
- Запланированное создание компонента с самого начала;
- Как расширение существующего компонента.

Два типа самописных компонентов

Я думаю, что вы уже заметили два пути их создания:

- Создание как наследника “**User Control**”;
- Создание как наследника “**Component**”.

Я использую наследование от класса Components, когда я хочу добавить новую функциональность к уже существующему компоненту без изменений во внешнем виде. Во всех других случаях я использую UserControl.

Большая часть всего ниже написанного применима к обоим типам.

Иконки в панели инструментов

Думаю, что и вам захочется поменять иконку своего компонента на что-то более привлекательное, чем изображение шестеренки. Основная часть работы у вас пройдет в графическом редакторе, если не в рисовании, так в подсчете использованных цветов. =)

Но давайте начнем с самого простого пути.

Когда вы создаете компонент и наследуете его от уже существующего, иконка будет наследована от базового класса.

```
public partial class VTLabel : Label { … }
```

После установки компонента в панель инструментов, вы увидите иконку Label.

Если же создание идет без наследования от стандартного компонента, но хотите использовать стандартную иконку вместо рисования своей, то на этот случай есть атрибуты. С помощью необходимого атрибута можно указать, иконку от какого базового компонента следует взять.

```
[ToolboxBitmap(typeof (Label))]
public partial class VTLabel : Component { … }
```

И как последний вариант, можете создать свою собственную уникальную иконку для вашего компонента. Для того, чтобы ее можно было применить, она должна соответствовать следующим критериям:

- 16 x 16 пикселей. В противном случае студия растянет иконку и результат будет ужасным;
- Использование только 16 цветов;
- Формат картинки BMP.

Когда все подготовлено, добавьте картинку к проекту. Выделите добавленную картинку и откройте окно свойств. Необходимо найти свойство **Build action** и изменить его на значение **Embedded resource.**

Если не хочется писать никакого кода, то назовите иконку точно так же как и компонент. Если же вы будете помещать иконки в отдельную проектную папку, либо хотите назвать иконку отлично от самого компонента, то необходимо воспользоваться атрибутом.

```
[ToolboxBitmap(typeof (VTLink), "LinkLabel.bmp")]
public partial class VTLink : UserControl { … }
```

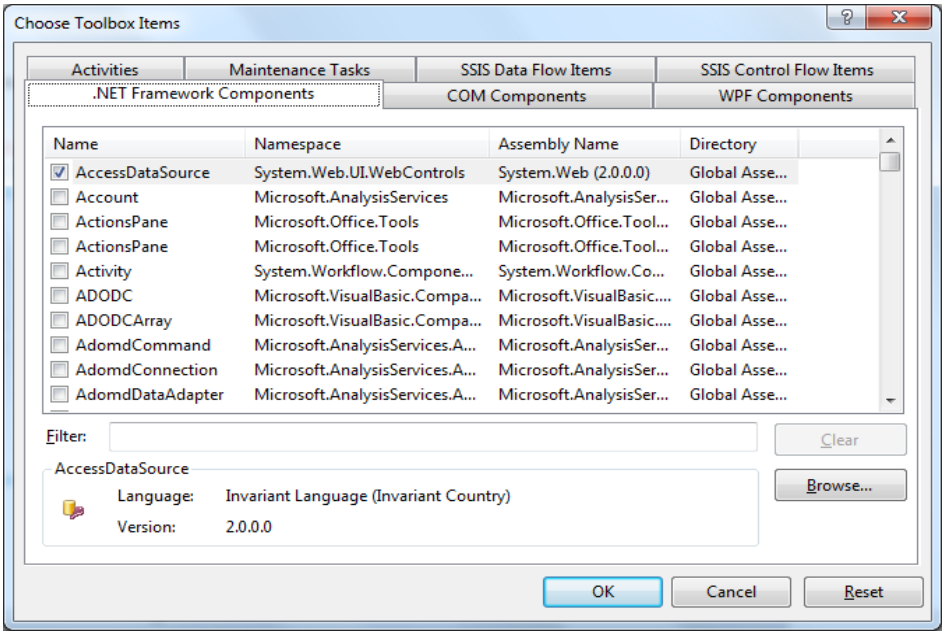
Или

```
[ToolboxBitmap(typeof (VTLink), "Resources.LinkLabel.bmp")]
public partial class VTLink : UserControl { … }
```

Во время разработки вы все равно будете видеть шестеренку. Для того, чтобы проверить, как оно будет выглядеть в реальной жизни, надо добавить компонент вручную.

Как добавить компонент вручную

Вызовите контекстное меню, щелкнув на пустом месте в панели инструментов. Выберите “Add new tab”. Введите любое имя. Снова вызовите контекстное меню, щелкнув по новой вкладке, и выберите “Choose Items…”. Вы должны увидеть окно как на картинке:



Нажмите кнопку “Browse…” и надите dll с необходимым компонентом. Если вы умудрились уже зарегистрировать библиотеку в GAC, то можно использовать фильтр и добавить компонент из текущего окна. Не забывайте, что окно инструментов показывается только те компоненты, которые возможно использовать для открытого документа.

Полезные атрибуты

Browsable

Говорит, надо ли показывать это свойство в окне свойств или нет.

```
[Browsable(true)]
public int Max { get {…} set{…} }
```

Bindable

Если вы планируете, что основное использование свойства будет для биндинга, то можно применить этот атрибут. Как результат использования, свойство появится в категории *Data* в секции “(DataBindings)”

```
[Bindable(true), Browsable(true)]
public string MainText { get {…} set{…} }
```

Category

Используйте этот атрибут, чтобы поместить свойство в определенную категорию.

```
[Category("Appearance"), Browsable(true)]
public Image Icon { get {…} set{…} }
```

Description

Предоставляет дополнительную информацию о свойстве. Все что тут напишете, появится внизу окна свойств.

```
[Description("The icon that appears on hover"), Browsable(true)]
public Image Icon { get {…} set{…} }
```

Hard’n’heavy!