Smart tag – Single Binding

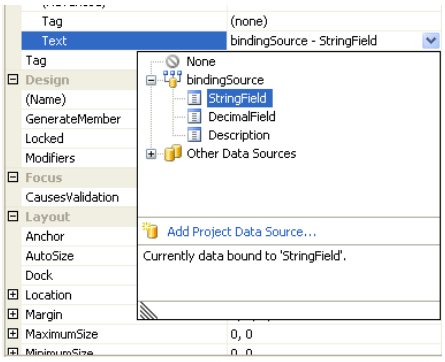
Прочитать на сайте

Intro

Продолжаем разговор на тему «Как создать смарт тэг». В предыдущем посте на эту тему я рассказал, что такое смарт тэги и с чем их едят. Сейчас же я хочу рассказать о том, как сделать биндинг для единичного поля.

В дальнейшем повествовании я буду исходить из предположения, что у вас уже есть свой компонент, к нему применены все заготовки для создания смарт тэга и все готово для дальнейших действий. Так же как и в прошлом посте, я советую использовать ReSharper.

Я собираюсь рассказать и показать, как создать такую же функциональность в смарт тэге, которую вы наблюдаете при задании биндинга в окне свойств.



Реализовав этот функционал, вы сможете сэкономить много времени, которое раньше тратилось на путешествия по окну свойств для того, чтобы настроить поведение компонента.

Есть ли стандартный способ задания биндинга?

Я его не нашел. Поиски «простого» и короткого стандартного способа не увенчались успехом. Sad, but true. Используя гугл, я нашел только 4 (четыре) вопроса по теме, и те без ответов. Только раз было направление для поисков, но для Web форм. Расстройству моему не было предела, но недолго оно длилось! Если кто-то что-то смог сделать, то другой человек может это сделать тоже. Руководствуясь этим железобетонным принципом, я вспомнил, что Microsoft опубликовала исходные коды для .Net фреймворка и потратив некоторое время на поиски я нашел где их слить и начал детальное исследование.

Используя свои знания по разработке компонентов и опыту работы с ними, я начал поиски с реализации ComboBox, чтобы посмотреть какой дизайнер используется для биндинга. К сожалению, там не оказалось какого-то общего дизайнера который можно было бы сразу прикрутить и получить профит. Но в целом это не было проблемой. Я нашел такие строчки:

```
[TypeConverter("System.Windows.Forms.Design.DesignBindingConverter"),
Editor("System.Windows.Forms.Design.DesignBindingEditor, System.Design",
typeof (UITypeEditor))]
```

Можете поискать в MSDN (http://social.msdn.microsoft.com/Search/en-US?query=DesignBindingEditor&ac=8) что пишут по поводу DesignBindingEditor в общем поиске и по коду. Что? А? Пишут что ничего нет?

Ну ладно, редактор и конвертор типов должны все равно быть стандартными и найти их проблемой быть не должно. Но это не все, иначе было бы слишком легко!))

Копи-паста

Это наш путь к пониманию сути вещей (если нет документации). Скопировать, вставить, немного менять код и смотреть как и что меняется. Да, метод научного тыка.

Как вы должны помнить, свойства, объявленные в классе дизайнера смарт тэга, будут потом использоваться для отображения пользователю и весь рабочий код помещается в “get”/“set”. Здесь много всего написано, что даже чуток пугает. =)

```
public object BoundSelectedValue {
            get {
                System.Windows.Forms.Binding b = GetSelectedValueBinding();
                string dataMember;
                object dataSource;
                if (b == null) {
                    dataMember = null;
                    dataSource = null;
                } else {
                    dataMember = b.BindingMemberInfo.BindingMember;
                    dataSource = b.DataSource;
                }
                string typeName =
string.Format(System.Globalization.CultureInfo.InvariantCulture,
"System.Windows.Forms.Design.DesignBinding, {0}",
typeof(ControlDesigner).Assembly.FullName);
                _boundSelectedValue = TypeDescriptor.CreateInstance(null,
Type.GetType(typeName), new Type[] { typeof(object), typeof(string) }, new
object[] { dataSource, dataMember });

                return _boundSelectedValue;

            }
            set {
                if (value is String) {
                    PropertyDescriptor pd =
TypeDescriptor.GetProperties(this)["BoundSelectedValue"];
                    TypeConverter tc = pd.Converter;
                    _boundSelectedValue = tc.ConvertFrom(new
EditorServiceContext(_owner),
System.Globalization.CultureInfo.InvariantCulture, value);
                } else {
                    _boundSelectedValue = value;
                    if (value != null) {
                        object dataSource =
TypeDescriptor.GetProperties(_boundSelectedValue)["DataSource"].GetValue(_bou
ndSelectedValue);
                        string dataMember =
(string)TypeDescriptor.GetProperties(_boundSelectedValue)["DataMember"].GetVa
lue(_boundSelectedValue);
                        SetSelectedValueBinding(dataSource, dataMember);
                    }
                }
            }
        }
```

Вау!

Начнем разбор полетов со свойства «get». В строке номер <ТАКОЙ-ТО> берутся все элементы составляющие компонент и ищутся биндинги с необходимыми именами. Для ComboBox это будет SelectedValue. Если ничего не найдено, значит биндинг еще не назначался на данное поле. Иначе, с помощью DesighnBindingConverter, берутся свойства dataMember и dataSource для создания строкового представления биндинга.

В результате экспериментов «get» оказался легкой частью для адаптации. Больше всего проблем возникло с «set». Если биндинг был уже назначен, то он в смарт тэге представлялся в виде строки, которую надо было разобрать (распарсить). Для этого использовалась первая часть выражения if. И в этой части, мне кажется и кроется причина почему же биндиг одиночного поля не опубликован в свободном доступе. В строке номер <ТАКОЙ-ТО> создается EditorServiceContext, который является внутренним классом в .Net. Придется его найти и «переписать» для наших нужд. Все переписывание конечно свелось к копи-пасте в локальный проект. Последнее выражение с SetSelectedValueBinding так же было скопировано без изменений.

Рефакторинг

Оставлять такую прорву кода как есть не хотелось, тем более трудно было бы создать еще одно свойство с биндингом. Так что я переделал код на более короткую запись. Получилось в итоге вот так:

```
[TypeConverter("System.Windows.Forms.Design.DesignBindingConverter"),
 Editor("System.Windows.Forms.Design.DesignBindingEditor, System.Design",
typeof (UITypeEditor))]
public object DefBindProperty {
    get { return GetBind("DefBind"); }
    set { SetBind("DefBind", value, "DefBindProperty"); }
}
```

Получилось более ясно и просто. Можно использовать не задумываясь для других компонентов. В данном случае первый параметр указывает на свойство в связанном со смарт тэгом компоненте.

GetBind и SetBind объявлены в классе ControlActionListBase. Э том случае ActionList наследуем от ControlActionListBase, а ControlActionListBase от DesignerActionList.

```
public class MyControlActionList : ControlActionListBase { … }

public class ControlActionListBase : DesignerActionList { … }
```

Таким образом, в нашем классе MyControlActionList остаются только значимый код для конкретного смарт тэга, что упростит дальнейшую редакцию и чтение. Весь служебный, общий код вынесен в базу. Написал и забыл. ;) Когда вы примените биндинг с помощью описанного кода через смарт тэг, в экземпляре компонента будет сгенерирован код похожий на этот:

```
this.myControl1.DataBindings.Add(new System.Windows.Forms.Binding("DefBind",
this.bindingSource, "StringField", true));
```

Все по-честному!

Финальный результат можно посмотреть в исходниках.

На данном этапе вы можете создавать свои компоненты и расширять уже существующие с биндингом на единичное поле, а не список. Я надеюсь, что это ускорит время разработки приложений.

Hard’n’heavy!