

Записки свободного админа..

OpsMgr, SCSM и все-все-все...

How-To: Создание собственного UserControl для SCSM 2010 SP1

01.07.2011 [2 КОММЕНТАРИЯ \(HTTPS://FREEMANRU.WORDPRESS.COM/2011/07/01/HOW-TO-USERCONTROL-FOR-SCSM-2010-SP1/#COMMENTS\)](https://freemanru.wordpress.com/2011/07/01/how-to-usercontrol-for-scsm-2010-sp1/#comments)

ВНИМАНИЕ! Описанные здесь техники в основном являются не документированными и не поддерживаемыми со стороны Microsoft. Информация предоставлена “как есть”, автор не несет ответственности за возможную потерю информации.

Вступление

Всё чаще я встречаю вопросы о том, как можно расширить функциональность SCSM с помощью собственных форм или контролов. Лично я предпочитаю не использовать полностью переписанные формы без крайней необходимости: это довольно большой объем работы, и при выходе новой версии продукта (выход намечен на Q4 2011 – Q1 2012) с почти 100% гарантией ваша форма перестанет работать. Контролы требуют меньше времени на разработку, а шанс, что они будут работать и в следующей версии продукта, довольно велик.

Чтобы понимать, о чем пойдет речь в данной статье, необходимо иметь представление о следующих технологиях:

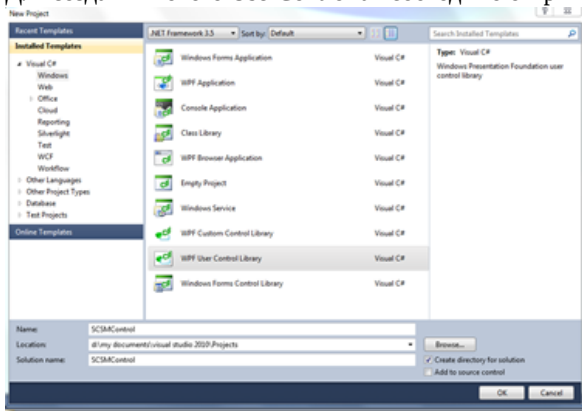
- WPF (Windows Presentation Foundation), в особенности Binding
- [DependencyProperty](http://msdn.microsoft.com/en-us/library/ms752914.aspx) (<http://msdn.microsoft.com/en-us/library/ms752914.aspx>)
- XML-схема пакета управления

Итак, нам необходимо решить следующие задачи:

1. Создать новый UserControl в Visual Studio
2. Подключить этот контрол к форме
3. Доставить библиотеку с контролом на все компьютеры с консолью SCSM

Создание UserControl-a

Для создания нового UserControl-a необходимо открыть Visual Studio, и выбрать новый проект **WPF User Control Library**:



(<https://freemanru.files.wordpress.com/2011/07/image.png>)

Имя решения и имя класса могут быть любыми. После этого необходимо отредактировать название контрола.

Затем необходимо подключить к проекту несколько библиотек:

1. **Microsoft.EnterpriseManagement.Core.dll** (расположена в папке c:\Program Files\Microsoft System Center\Service Manager 2010\SDK Binaries на сервере SCSM)
2. **Microsoft.EnterpriseManagement.UI.Foundation.dll** (расположена в папке c:\Program Files\Microsoft System Center\Service Manager 2010\)
3. **Microsoft.EnterpriseManagement.UI.SdkDataAccess.dll** (расположена в папке c:\Program Files\Microsoft System Center\Service Manager 2010\)

Если вы планируете использовать стандартные контролы SCSM вам также понадобится подключить библиотеки

Microsoft.EnterpriseManagement.UI.Controls.dll, **Microsoft.EnterpriseManagement.UI.ExtendedControls.dll**, **Microsoft.EnterpriseManagement.UI.SMControls.dll** и **WPFToolkit.dll**

Все контролы, которые мы подключаем через расширение форм, обязаны иметь атрибут ContentProperty (<http://msdn.microsoft.com/en-us/library/system.windows.markup.contentpropertyattribute.aspx>). С чем именно связано такое требование, мне выяснить не удалось, но такое требование обязательно. В связи с этим нам необходимо создать переменную, которую мы будем использовать в качестве значения для атрибута ContentProperty. Тип и название переменной могут быть любыми, но она должна быть определена с помощью DependencyProperty (<http://msdn.microsoft.com/en-us/library/ms752914.aspx>).

В итоге у вас должно получиться нечто, похожее на это:

```

01 namespace SCSMControls
02 {
03     [ContentProperty("SelectedItem")]
04     public partial class SCSMControl : UserControl, INotifyPropertyChanged
05     {
06         public event PropertyChangedEventHandler PropertyChanged;
07
08         public SCSMControl()
09         {
10             InitializeComponent();
11         }
12
13         public static readonly DependencyProperty SelectedItemProperty =
14             DependencyProperty.Register("SelectedItem", typeof(string),
15             typeof(SCSMControl), new PropertyMetadata(null));
16
17         public string SelectedItem
18         {
19             get
20             {
21                 return (string)base.GetValue(SelectedItemProperty);
22             }
23             set
24             {
25                 base.SetValue(SelectedItemProperty, value);
26                 NotifyPropertyChanged("SelectedItem");
27             }
28         }
29
30         private static void OnSelectedItemChanged(DependencyObject sender,
31             DependencyPropertyChangedEventArgs e)
32         {
33             //TODO
34         }
35
36         /// <summary>
37         /// INotifyPropertyChanged implementation
38         /// </summary>
39         /// <param name="propertyName"></param>
40         private void NotifyPropertyChanged(string propertyName)
41         {
42             if (this.PropertyChanged != null)
43             {
44                 this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
45             }
46         }
47     }
48 }

```

Я также добавил интерфейс `INotifyPropertyChanged`, чтобы внешние компоненты могли подписываться на изменение свойств нашего контрола. Кроме этого, я показал как можно подписаться на изменение свойства нашего компонента. Иногда это бывает полезно. Но использовать оба эти подхода не обязательно.

Теперь наш контрол можно добавлять на форму в SCSM, но он пока не делает ничего полезного. Для начала нужно отобразить какие-нибудь данные. В этом нам поможет такой мощный механизм WPF, как привязка (binding). Но чтобы использовать привязку, необходимо знать к каким свойствам привязываться.

`DataContext` формы (а значит и нашего контрола) заполняется объектом типа `IDataItem`. Этот тип не документирован, поэтому вам придется исследовать его самим или поверить мне)). Объект типа `IDataItem` хранит в себе все свойства объекта, для которого открыта форма. Доступ к объектам осуществляется операцией взятия индекса (`[]`), а индексатором выступает внутреннее имя поля (или имя `Type Projection`). Несколько примеров:

```
1 | IDataItem item = this.DataContext as IDataItem;
2 | string title = (string)item["Title"];
3 | string status = (string)(item["Status"] as IDataItem)["Dis
4 | string affectedUser = (string)(item["AffectedUser"] as IDaI
```

Как видно из примера, мы можем обращаться к вложенным свойствам полученных объектов также через `IDataItem`. Чтобы использовать эти значения в привязке, достаточно указать название свойства. Добавим на наш контрол несколько элементов:

```
1 | <TextBox Name="boxID" Grid.Column="1" Grid.Row="0" Text="{
2 | <TextBox Name="boxName" Grid.Column="1" Grid.Row="1" Text=
3 | <TextBox Name="boxAffectedUser" Grid.Column="1" Grid.Row="
4 |     <TextBox.Text>
5 |         <Binding Path="AffectedUser.DisplayName" Mode="OneWay"
6 |     </TextBox.Text>
7 | </TextBox>
8 | <TextBox Name="boxItem" Grid.Column="1" Grid.Row="3" Text=
```

Здесь показаны несколько приемов привязки. В первом случае мы привязываемся к внутреннему идентификатору элемента (тип `Guid`). Во втором случае – к стандартному. Третий способ показывает, как можно задать текст для случая, если значение поля пустое. Заметьте, что это не значение по-умолчанию, а именно подстановка текста. Четвертый способ показывает, как можно сделать привязку к свойствам нашего контрола.

Итак, наш контрол умеет отображать данные. Но не плохо бы научить его также и изменять данные. `IDataItem` не слишком хорошо подходит для манипуляции с объектами SDK, т.к. он содержит лишь информацию об одном объекте. Не плохо бы получить доступ к SDK, т.к. с помощью него мы можем производить любые манипуляции с данными. Это мы можем сделать следующим образом (см. функцию `GetSession`):

```

1  [ContentProperty("SelectedItem")]
2  public partial class SCSMControl : UserControl, INotifyPropertyChanged
3  {
4      EnterpriseManagementGroup mg;
5
6      public event PropertyChangedEventHandler PropertyChanged;
7
8      public SCSMControl()
9      {
10         InitializeComponent();
11         GetSession();
12     }
13
14     void GetSession()
15     {
16         // Get the current session, more info: http://msdn.microsoft.com/en-us/library/ff634631.aspx
17         IServiceContainer container = (IServiceContainer)Application.Current;
18         IManagementGroupSession curSession = (IManagementGroupSession)container.GetService(typeof(IManagementGroupSession));
19         if (curSession == null)
20             throw new ValueUnavailableException("curSession is null");
21         mg = curSession.ManagementGroup;
22     }
23 }

```

Итак, мы имеем доступ к SDK, давайте сделаем что-нибудь полезное. Н-р установим свойства по-умолчанию для нового объекта. Для этого нам необходимо получить объект IDataItem из DataContext, а затем установить свойства. Делать это в обработчике события FormLoaded не стоит – в этом момент DataContext еще не заполнен. Вместо этого мы подписываемся на изменение свойства DataContext, и когда там оказывается нужный нам объект – устанавливаем свойства:

```

1  private void UserControl_DataContextChanged(object sender,
2  {
3      // wait binding
4      if (this.DataContext is IDataItem)
5      {
6          instance = (this.DataContext as IDataItem);
7          // If this is new incident, set some default properties
8          if ((bool)instance["$IsNew$"])
9          {
10             instance["Title"] = "WOW! Now we can set the title";
11             instance["Description"] = "And we can use SDK";
12             //IncidentTierQueuesEnum.Tier2
13             instance["TierQueue"] = mg.EntityTypes.GetEnum(typeof(IncidentTierQueuesEnum), "Tier2");
14         }
15     }
16 }

```

Обратите внимание на свойство \$IsNew\$ – оно определяет открыта ли форма для создание элемента (true) или для редактирования (false).

Итак, на этом наш компонент полностью готов. Проект Vusial Studio 2010 с примером вы можете скачать в конце статьи.

Добавление контрола на форму

Теперь нам необходимо добавить наш контрол на форму. Для этого нам потребуется создать с помощью Authoring Tool новую модификацию для формы, а затем отредактировать её XML-код в любом редакторе. Чтобы добавить собственный контрол на форму необходимо:

1. Создать новую модификацию формы. Как это сделать описано много где в сети, например [здесь](http://blogs.technet.com/b/servicemanager/archive/2011/01/14/customizing-forms-part-1-adding-a-new-user-picker-control-to-the-incident-form.aspx) (<http://blogs.technet.com/b/servicemanager/archive/2011/01/14/customizing-forms-part-1-adding-a-new-user-picker-control-to-the-incident-form.aspx>) и [здесь](http://blogs.technet.com/b/servicemanager/archive/2011/01/14/customizing-forms-part-2-adding-the-affected-user-picker-to-the-change-request-form.aspx) (<http://blogs.technet.com/b/servicemanager/archive/2011/01/14/customizing-forms-part-2-adding-the-affected-user-picker-to-the-change-request-form.aspx>), а вот [здесь](http://blogs.technet.com/b/servicemanager/archive/2010/09/27/adding-a-customer-property-to-the-incident-class-and-form.aspx) (<http://blogs.technet.com/b/servicemanager/archive/2010/09/27/adding-a-customer-property-to-the-incident-class-and-form.aspx>) даже с видео
2. Добавить на форму в место, где должен находится наш контрол, любой стандартный контрол, например Label
3. Сохранить пакет управления, затем открыть его в любом текстовом редакторе.
4. Найти секцию Forms, а в ней добавленный контрол.
5. Заменить атрибуты Assembly и Type на данные нашего контрола. PublicKey можно узнать с помощью команды sn.exe -T <путь к сборке> (находится в папке c:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\). Можно добавить эту команду (<http://blogs.msdn.com/b/miah/archive/2008/02/19/visual-studio-tip-get-public-key-token-for-a-strong-named-assembly.aspx>) в инструменты Visual Studio
6. При необходимости, добавить другие свойства с помощью элемента <PropertyChange>

Вот пример для формы инцидента:

```

1  <Form ID="CustomForm_650994d4_1a69_4f1a_975f_7d060b90a3f3"
2  <Category>Form</Category>
3  <Customization>
4    <AddControl Parent="StackPanel206" Assembly="Presentati
5    <PropertyChange Object="Label_1" Property="HorizontalAl:
6      <NewValue>Left</NewValue>
7    </PropertyChange>
8    <PropertyChange Object="Label_1" Property="VerticalAlign
9      <NewValue>Bottom</NewValue>
10   </PropertyChange>
11   <PropertyChange Object="Label_1" Property="Content">
12     <NewValue>Custom control:</NewValue>
13   </PropertyChange>
14   <AddControl Parent="StackPanel206" Assembly="SCSMContro
15   <PropertyChange Object="SCSMControl_1" Property="Width":
16     <NewValue>Auto</NewValue>
17   </PropertyChange>
18   <PropertyChange Object="SCSMControl_1" Property="Height"
19     <NewValue>Auto</NewValue>
20   </PropertyChange>
21   <PropertyChange Object="SCSMControl_1" Property="Vertic
22     <NewValue>Bottom</NewValue>
23   </PropertyChange>
24   <PropertyChange Object="SCSMControl_1" Property="Margin"
25     <NewValue>0,0,0,0</NewValue>
26   </PropertyChange>
27   <PropertyChange Object="SCSMControl_1" Property="Select
28     <NewValue>Cool control</NewValue>
29   </PropertyChange>
30 </Customization>
31 </Form>

```

В итоге у меня получился вот такой контрол:

Custom contro	
Element ID:	42b12ebd-d498-1611-e27a-70f31ec1109f
Element Name:	IR1275
Affected User:	Антон М. Гриценко
Selected Item:	Cool control

(<https://freemanru.files.wordpress.com/2011/07/image1.png>)

Для новых инцидентов автоматически заполняется Название и Группа поддержки:

(<https://freemanru.files.wordpress.com/2011/07/image2.png>).

Доставка библиотеки

Теперь нам необходимо скопировать нашу библиотеку на все рабочие станции с консолью SCSM. Хорошо, если их 3-4, а если их 50?100? А как потом обновлять эту библиотеку? Не самые приятные и простые вопросы.

К счастью для нас, разработчики SCSM позаботились об этом. В SCSM существует так называемый бандл пакетов управления (management pack bundle (<http://blogs.technet.com/b/service manager/archive/2009/09/04/introducing-management-pack-bundles.aspx>)). Данный тип пакетов управления может содержать в себе другие пакеты управления (как запечатанные, так и нет), а также различные сборки, изображения и прочие ресурсы.

Необходимо лишь перед созданием бандла в пакете управления указать ссылку на ресурс (в нашем случае библиотека). Добавленные таким образом ресурсы копируются в локальный профиль пользователя, который запускает консоль, в папку `%USERPROFILE%\AppData\Local\Microsoft\System Center Service Manager 2010\%GROUPNAME%\%MPVERSION%`,

где

`%GROUPNAME%` – имя группы управления

`%MPVERSION%` – версия пакета управления, который содержит ссылку на ресурс

Нам достаточно добавить в конце нашего пакет управления ссылку на сборку:

```
1 </LanguagePacks>
2 <!-- Section For Assembly -->
3 <Resources>
4   <Assembly ID="SCSMControlAssembly" Accessibility="Publ:
5   </Resources>
6 </ManagementPack>
```

а затем упаковать его в бандл. Для упаковки вы можете использовать скрипт по ссылке выше или мою утилиту MPBMaker (<http://gallery.technet.microsoft.com/MPB-Maker-743f85b9>) (перед упаковкой не забудьте скопировать библиотеку в ту же папку, где расположен пакет управления):

MPBMaker.exe SCSMControlBundle "d:\Examples\Example.SCSMControl.xml"

Полученный пакет необходимо импортировать в SCSM.

Возникающие ошибки

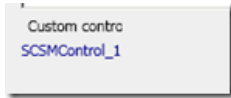
Если во время импортирования пакета управления вы получили ошибку вроде этой:

: Failed to verify form: CustomForm_650994d4_1a69_4f1a_975f_7d060b90a3f3

The form base is not valid. Form CustomForm_650994d4_1a69_4f1a_975f_7d060b90a3f3 extends form System.WorkItem.Incident.ConsoleForm, which already has another extension (CustomForm_8c4ec25b_1dc3_4b58_bd43_7c8a83f619a0)

то это означает, что форма уже модифицирована в другом пакете управления.

Если после импортирования пакета ваш контрол выглядит вот таким образом:



(<https://freemanru.files.wordpress.com/2011/07/image3.png>)

то скорее всего вы забыли добавить атрибут ContentProperty.

Заключение

С помощью собственных контролов вы можете полностью контролировать поведение формы:

- Задавать значение по-умолчанию для свойств
- Изменять параметры других контролов (н-р отключать или включать обязательность полей)
- Отключать или прятать другие контролы на основе каких-то параметров или роли пользователя

Скачать готовый проект с примером на Visual Studio, а также готовый пакет управления вы можете [здесь](https://skydrive.live.com/?cid=9e1589588902dbaa&sc=documents&uc=2&id=9E1589588902DBAA%211040#) (<https://skydrive.live.com/?cid=9e1589588902dbaa&sc=documents&uc=2&id=9E1589588902DBAA%211040#>).

РУБРИКА: SCSM TAGGED WITH ПРОГРАММИРОВАНИЕ, ФОРМЫ, SCSM, WPF

2 Responses to *How-To: Создание собственного UserControl для SCSM 2010 SP1*

SigmuS says:

02.02.2012 в 07:46

Очень познавательная статья. Особенно порадовала возможность делать обязательными другие поля на форме, например «Описание», бывает полезным. Есть один вопросик — после кастомизации МП можно ли его запечатать, потому что незапечатав его, не получится применять к расширенной форме шаблоны.

Ответить

freemanru says:

02.02.2012 в 13:27

Конечно можно.

Ответить

Блог на WordPress.com.