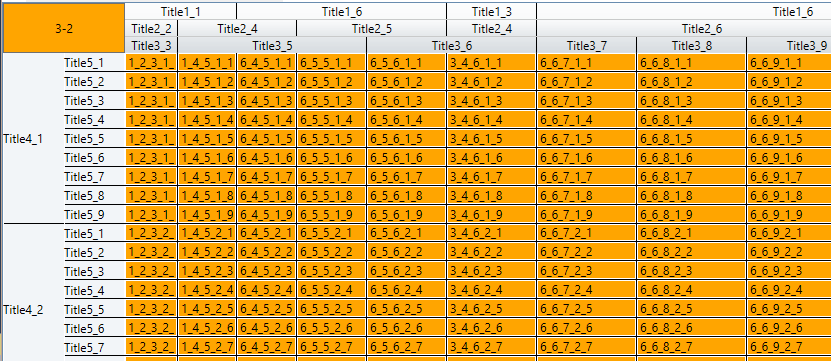
1.Введение

Стандартные средства представления данных в табличной форме в .NET не очень хорошо приспособлены для многомерных данных. Для этой цели используются специализированные средства. Но интерес к представлению таких данных именно стандартными средствами .NET существует, например [1-8]. В настоящей статье предлагается один из способов представления многомерных данных в WPF DataGrid.



Перераспределяя данные между строками и столбцами и соответствующим образом сортируя их можно существенно влиять на их «удобочитаемость».

Предлагаемый способ основан на взаимодействии с компонентами VisualTree DataGid такими как список заголовков столбцов, заголовков строк, вертикальный и горизонтальный ScrollBar, область скроллинга ячеек ScrollViewer и другими. Используется стандартный DataGrid DefaultTemplate из PresentationFramework 4.0 (4.6.1055.0). Не все используемые компоненты в DefaultTemplate именованы. При использовании другого Template может потребоваться модификация методов доступа к компонентам VisualTree DataGid реализованных в классе DataGridExHelper.

В тестовой программе приведен пример использования в MVVM.

2.Модель

Используем широко распространенную модель «[Звезда](https://en.wikipedia.org/wiki/Star_schema)». В этой модели факт можно представить функцией от нескольких аргуметов – значений измерений (Dimensions), где N – это количество измерений

w = Fact(arg0, arg 1, …, argN).

Представим это в виде интерфейса:

interface IModel{

IDimension[] Dimensions {get;}

Object Fact(IDimItem[] arg);

}

interface IDimension{

int DimId {get;}

String DimName {get;}

IDimItem[] DimItems {get;}

}

interface IDimItem{

int DimId {get;}

UInt64 DimItemId {get;}

String DimItemName {get;}

}

Каждый факт является функцией от нескольких аргуметов типа IDimItem. Если какой-то из аргументов имеет фиксированное значение (например если DimItems.Length <= 1), то, в случае табличного представления фактов, такое Dimension может быть вынесено за пределы таблицы и количество Dimensions, отражаемых в таблице, может быть уменьшено. Обозначим это количество Dimensions через M, M<=N.

3.Как данные Модели преобразуются в таблицу

Факт отображается в ячейке таблицы. Не обязательно каждой ячейке таблицы соответствует факт. Все M Dimensions подразделяются на две группы: DimensionsRow для отображения в строках и DimensionsCol для отображения в столбцах. Каждой строке таблицы соответствует комбинация IDimItem из DimensionsRow. Каждому столбцу таблицы соответствует комбинация IDimItem из DimensionsCol.

Если M = 0, то наша таблица вырождается в одну ячейку.

Если M = 1, то мы имеем таблицу либо с одной строкой, либо с одним столбцом.

Если M = 2, то мы можем иметь «нормальную» двумерную таблицу.

Если M > 2, то в определении строк и|или столбцов должны присутствовать комбинации значений нескольких IDimItem.

4.Что сделано

Класс DataGridEx унаследован от System.Windows.Controls.DataGrid и позволяет отображать многомерные данные в табличной форме как это показано на рисунке. Все изменения касаются только отображения заголовков строк и столбцов. Класс имеет дополнительные свойства:

IEnumerable ViewItemsSource - выполняет функции свойства ItemsSource стандартной DataGrid;

Style TopLeftHeaderStyle - стиль самой верхней ячейки в столбце заголовков строк;

Bool MergingHeadersCol - объединение (Merging) заголовков столбцов;

Bool MergingHeadersRow - объединение (Merging) заголовков строк;

Дополнительные классы ViewItemsSource, ViewTypePass, ViewPropPass, ViewItemPass предназначены для формирования вида таблицы в терминах модели.

5.Как это можно использовать

5.1.ViewModel

Создание ViewModel определяется четырьмя классами.

ViewTypePass задает общие характеристики таблицы:

- какие Dimensions вынесены за пределы таблицы и какие DimItems в них

выбраны;

- как распределены оставшиеся Dimensions между строками и столбцами;

- какие первоначальные размеры выбраны для заголовка строки;

ViewTypePass typePass = new ViewTypePass();

...

typePass.FixedDims = new ulong[0];

typePass.FixedIds = new ulong[0];

typePass.ColDims = new ulong[3] {1, 2, 3};

typePass.RowDims = new ulong[2] {4, 5};

typePass.RowHeaderPartWidth = new double[2]{60, 60 };

ViewPropPass определяет столбец:

- тип данных в столбце;

- какие IDimItem из DimensionsCol выбраны для столбца;

- первоначальный размер столбца;

ViewPropPass propPass = new ViewPropPass(...,

typeof(int),

new UInt64[] { 101, 202, 303 },

40,

ColumnWidthType.Fixed);

ViewItemPass определяет строку таблицы:

- какие IDimItem из DimensionsRow выбраны для строки;

ViewItemPass itemPass = new ViewItemPass(...,

new UInt64[] { 404, 505 });

ViewItemsSource объединяет в себе все компоненты ViewModel:

- общие характеристики таблицы (ViewTypePass);

- ссылка на модель;

- список столбцов (ViewPropPass);

- список строк (ViewItemPass);

IModel model = null;

ViewItemsSource coll = null;

...

coll = new ViewItemsSource(model, typePass);

...

coll.ItemPropAdd(propPass);

...

coll.ItemAdd(itemPass);

Именно экземпляр класса ViewItemsSource ставится в соответствие свойству

DataGridEx.ViewItemsSource

dataGridEx.ViewItemsSource = coll;

В тестовой программе представлен пример создания класса ViewModel в котором в зависимости от внешних условий выбирается тот или иной вариант построения таблицы, а результат представлен свойством ItemsSource.

public class ViewModel:INotifyPropertyChanged {

private IModel \_model = null;

private ViewItemsSource \_coll = null;

...

public ViewModel(IModel model) {

\_model = model;

...

}

public ViewItemsSource ItemsSource {get{return \_coll;}}

...

private void OnComm3\_2 (object par) {

ViewTypePass typePass = new ViewTypePass();

ViewPropPass propPass = null;

ViewItemPass itemPass = null;

...

\_coll = new ViewItemsSource(\_model, typePass);

...

coll.ItemPropAdd(propPass);

...

coll.ItemAdd(itemPass);

...

NotifyPropertyChanged("ItemsSource");

}

...

}

5.2.View

Пример привязки к ViewModel

public partial class MainWindow:Window {

public MainWindow() {

IModel model = null;

ViewModel vm = null;

...

vm = new ViewModel(model);

DataContext = vm;

}

...

}

<Window x:Class="VK\_De.WPF.NDimInDg.Test.MainWindow"

...

xmlns:my="clr-namespace:VK\_De.WPF.NDimInDg;assembly=NDimInDg"

Title="MainWindow" WindowState="Maximized">

...

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30" />

<RowDefinition />

<RowDefinition Height="30" />

</Grid.RowDefinitions>

...

<my:DataGridEx Grid.Row ="1"

...

ViewItemsSource ="{Binding ItemsSource}"

... />

...

</Grid>

</Window>

6.Как это функционирует

6.1.DataGridEx.ViewItemsSource

Присвоение значения свойству DataGid.ItemsSource должно быть синхронизировано, для корректной работы с компонентами VisualTree DataGid, с завершением загрузки экземпляра DataGid и выполнением некоторых предварительных действий. А свойство DataGid.ItemsSource не является виртуальным. Поэтому в DataGridEx создано дополнительное свойство ViewItemsSource, **которое и должно принять список отображаемых объектов**.

DataGridEx.ViewItemsSource принимает в качестве значения объект типа ViewItemsSource унаследованный от ObservableCollection <ViewItem>.

Поиск необходимых компонентов в VisualTree DataGrid делается методами класса DataGridExHelper при обработке события Loaded.

ManualResetEvent \_isReady используется для синхронизации с событием Loaded в методе BindingFromAnotherThread (...).

Переопределенное DataGridEx.ItemsSource:

- выбирает с помощью Lock соответствующий ViewItemTypeDescriptor;

- определяет размерность заголовков строк и столбцов;

- выбирает шаблоны для заголовков;

- делает стандартное присвоение base.ItemsSource = ...;

- активирует перерисовку заголовков строк и столбцов.

6.2.Модель и DataGrid.ItemsSource

Для каждого ViewItem DataGrid создает строку и каждый ViewItem имеет свойство Pass.IdsItem содержащее комбинацию IDimItem.DimItemId из DimensionsRow для строки.

В случае AutoGenerateColumns = true, DataGrid автоматически создает столбцы, используя TypeDescriptor для объектов из списка ItemsSource. В нашем случае для объектов типа ViewItem мы используем собственный ViewItemTypeDescriptor, который, посредством списка ViewItemPropertyDescr, передает в DataGrid список свойств, фактически не существующих во ViewItem, но соответствующих желаемым комбинациям IDimItem из DimensionsCol.

Этот список формируется функцией

ViewItemsSource.ItemPropAdd(ViewPropPass p),

где экземпляр класса ViewPropPass, помимо прочих свойств содержит в IdsProp комбинацию IDimItem.DimItemId для столбца. Таким образом мы вынуждаем DataGrid создать необходимый нам список столбцов.

Сложность при этом заключается в том, что для одного и того же типа ViewItem, но для разных ViewItemsSource, метод ViewItemTypeDescriptionProvider.GetTypeDescriptor(Type objectType, object instance) вызывается как в момент создания ViewItemsSource, так и в процессе присвоения base.ItemsSource = … и во втором случае он должен возвратить ViewItemTypeDescriptor соответствующего ViewItemsSource. Это достигается синхронизацией Lock.

6.3.Виртуализация строк

DataGrid поддерживает ограниченное количество строк. Взаимосвязь между элементом ViewItemsSource и строкой DataGrid устанавливается в методе OnLoadingRow и уничтожается в методе OnUnloadingRow.

6.4.Отображение данных в ячейках DataGrid

В момент загрузки строки DataGrid обращается к методу ViewItemPropertyDescr. GetValue(…) каждого из свойств-столбцов, передавая ему соответствующий ViewItem. В этот момент мы знаем все необходимые параметры для метода IModel.Fact(…). Если факт существует, то метод ViewItemPropertyDescr. GetValue(…) возвращает его значение и DataGrid отображает его в ячейке, в противном случае ячейка остается пустой.

6.5.Отображение заголовков строк и столбцов

Объединение (Merging) заголовков является важным моментом, оно может существенно улучшить «читаемость» таблицы и управляется свойствами DataGridEx.MergingHeadersCol и DataGridEx.MergingHeadersRow. Методы реализующие Merging сосредоточены в регионах «Refresh Headers Column» и «Refresh Headers Row».

7.Ссылки:

1[. Customizing Silverlight 3 DataGrid Headers в блоге Dan Wahlin's.](http://weblogs.asp.net/dwahlin/customizing-silverlight-3-datagrid-headers)

2. [GridView column header merging in ASP.NET](http://www.codeproject.com/Articles/448762/GridView-column-header-merging-in-ASP-NET)

3. [Silverlight Answers. Dynamic Multiple row header in DataGrid](http://www.go4answers.com/Example/dynamic-multiple-row-header-datagrid-154291.aspx)

4. [DataGridControl: Multiple Header Rows](https://social.msdn.microsoft.com/Forums/silverlight/en-US/8fb31c32-d98d-40c5-8ad1-72afde118504/datagrid-control-multiple-header-rows?forum=silverlightcontrols)

5. [Pivoting Datatable Simplified](Pivoting%20Datatable%20Simplified%20of%20Anurag%20Gandhi)

6. [Merge DataGrid Header](https://social.msdn.microsoft.com/Forums/vstudio/en-US/3fcbdaa3-17ae-41e9-b9bb-321d45db7f70/merge-datagrid-header?forum=wpf)

7. [Dynamic Multiple Row Column Grid Header](http://www.codeproject.com/Articles/27824/Dynamic-Multiple-Row-Column-Grid-Header)

# 8. [WPF DataGrid слияния заголовков](http://www.cyberforum.ru/wpf-silverlight/thread953690.html)