Приложения для Windows 8

4

Все приложения для Windows 8 обладают общими свойствами, которые позволяют им отлично работать на разных устройствах, на которых установлена эта операционная система. Это быстрые и гибкие приложения, которые легко адаптируются к различным режимам просмотра и экранам различных размеров. С ними можно взаимодействовать, используя различные способы ввода данных. Они поддерживают сенсорное взаимодействие с пользователем и предоставляют ему единообразный интерфейс, в котором для управления приложением предусмотрены стандартные места размещения команд и контекстных меню.

В предыдущей главе вы познакомились с XAML, декларативным языком описания разметки, который позволяет отделить дизайн от программирования, что дает возможность создавать мощные и расширяемые пользовательские интерфейсы. В этой главе вы узнаете, как посредством XAML-разметки и исполняемого программного кода наделить приложения для Windows 8 разными уникальными возможностями. Причем многие из этих возможностей доступны для использования и настройки непосредственно из XAML-разметки.

Макеты и режимы просмотра

Приложения для Windows 8 могут работать с различными макетами и режимами просмотра. В частности, доступны различные ориентации устройства (портретная и альбомная), а также состояния фиксации и заполнения,

когда окна двух приложений делят экран, располагаясь рядом друг с другом. Встроенные шаблоны Visual Studio 2012 дают приложениям возможность переключаться между различными режимами просмотра автоматически. Воспользовавшись встроенным имитатором, можно без проблем протестировать механизм смены макета приложением, даже если у вас нет планшетного компьютера или акселерометра.

Имитатор

Откройте пример Windows8Application к главе 4 этой книги. Если вы еще не загрузили примеры, их можно найти на странице http://windows8applications.codeplex.com/.

Вместо того чтобы начинать отладку приложения на локальном компьютере, как это происходит по умолчанию, выберите в раскрывающемся списке режимов отладки вариант Simulator (имитатор), как показано на рис. 4.1.

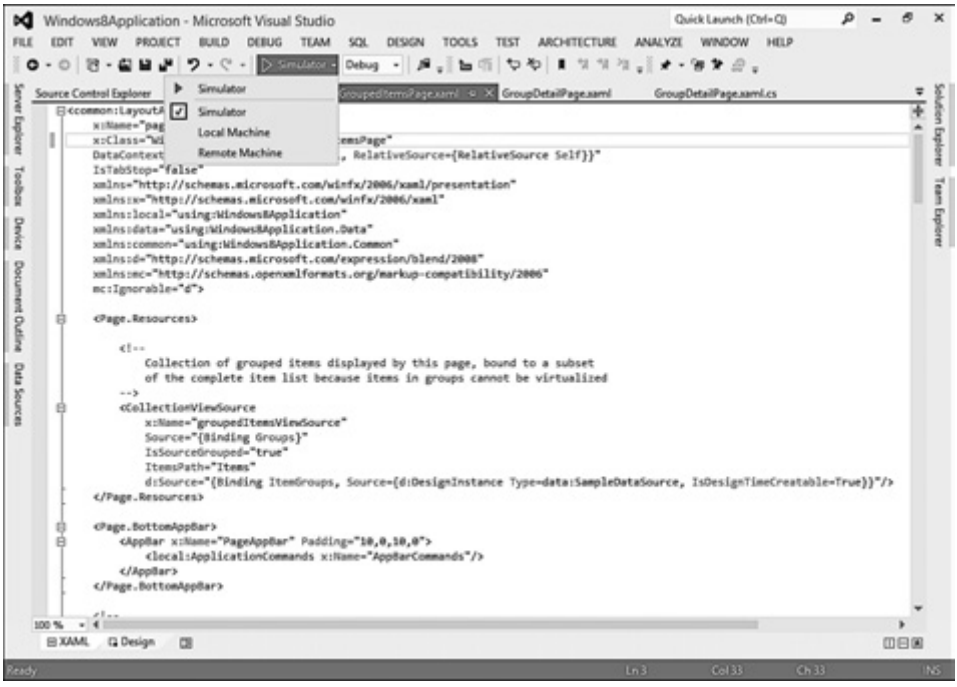

Рис. 4.1. Подготовка к отладке приложения в имитаторе

Имитатор упрощает тестирование различных вариантов использования приложения, таких как выполнение на экранах с разным разрешением и в различных режимах просмотра. Кроме того, он предоставляет имита-

цию сенсорного экрана в том случае, если вы занимаетесь разработкой приложения на компьютере, дисплей которого не является сенсорным. Окно имитатора оформлено в стиле, делающем его похожим на планшетный компьютер с клавишей Windows. С правой стороны окна имитатора расположен набор кнопок для доступа к различным функциям, в том числе (сверху вниз):

1. Mouse Mode (режим мыши). Позволяет использовать указатель мыши обычным способом.
2. Touch Emulation (базовый сенсорный режим). Имитация сенсорного взаимодействия с устройством с помощью мыши. Для того чтобы выполнить жест касания, достаточно щелкнуть в нужном месте левой кнопкой мыши.
3. Touch Emulation Pinch and Zoom (сенсорный режим с имитацией жестов масштабирования). В этом режиме можно протестировать жесты масштабирования (сжатия и расширения). Для их выполнения достаточно нажать левую кнопку мыши и, не отпуская ее, вращать колесо мыши для увеличения или уменьшения масштаба. То же самое можно выполнить и в режиме мыши (Mouse Mode), удерживая нажатой клавишу Ctrl и вращая колесо мыши.
4. Touch Emulation Rotate (сенсорный режим с имитацией жеста вращения). Режим имитации вращения реализуется удержанием левой кнопки мыши и вращением ее колеса.

Rotate 90 Degrees Clockwise (вращение по часовой стрелке на 90 градусов). Эта команда позволяет повернуть окно имитатора на 90° по часовой стрелке. В результате меняется ориентация экрана имитируемого устройства.

Rotate 90 Degrees Counterclockwise (вращение против часовой стрелки на 90 градусов). Данная команда осуществляет поворот окна имитатора на 90 градусов против часовой стрелки. Аналогично предыдущей команде, это приводит к смене ориентации экрана имитируемого устройства.

1. Change Resolution (смена разрешения). Позволяет менять разрешение экрана имитируемого устройства. С помощью этой команды можно установить разрешение экрана виртуального устройства даже большим, чем физическое разрешение дисплея, на котором вы работаете.
2. Set Location (задание положения). С помощью этой команды можно задать географическое местоположения имитируемого устройства, установив широту и долготу, высоту над уровнем моря и погрешность измерений.
3. Screenshot (получение снимка экрана). Эта удобная команда позволяет делать снимки экрана имитатора в его текущем разрешении.

4. Configure Screenshot (настройка снимка экрана). С помощью данной команды можно указать, нужно ли помещать копии экрана имитатора в буфер обмена или сохранять в виде файлов по заданному пути.
5. Help (справка). Вывод справочной информации об имитаторе.

Имитатор обладает богатыми возможностями. Установите разрешение окна имитатора в 1024×768. Вы увидите обычное приложение, содержащее элементы и группы элементов, как показано на рис. 4.2. Действие имитатора основано на подключении удаленного рабочего стола к устройству, на котором вы исполняете приложение (что отличается от принципов работы виртуальной машины). Это позволяет вам в полной мере протестировать приложение на собственном компьютере, применяя различные варианты сенсорного взаимодействия с приложением и различные параметры дисплея.

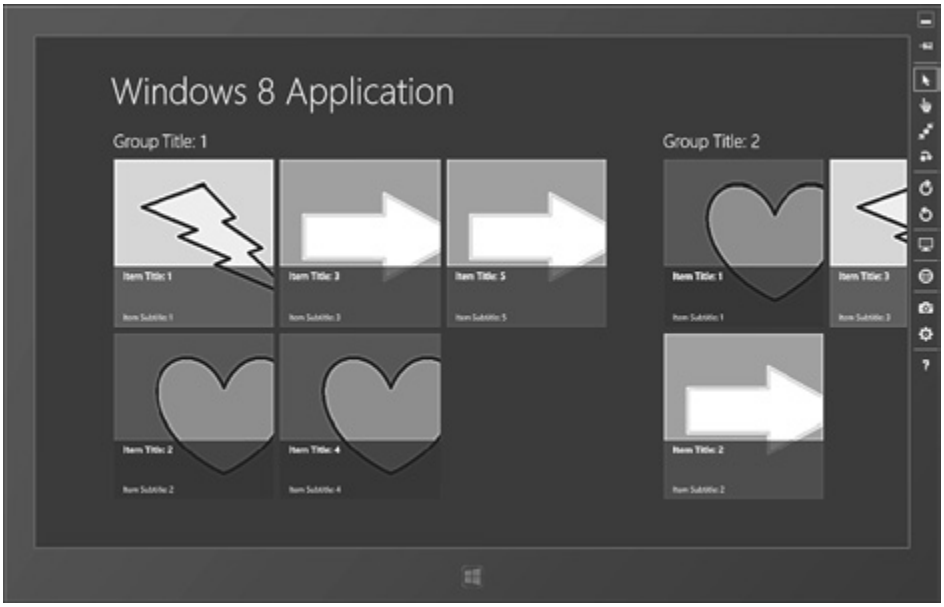

Рис. 4.2. Приложение запущено в имитаторе

Убедившись в том, что приложение имеет фокус ввода (вы можете щелкнуть мышью в свободном пространстве окна приложения или коснуться его, чтобы гарантировать, что это так), удерживая клавишу Windows, нажмите клавишу . (точка). Это заставит приложение перейти в режим фиксации. Если при этом исполняется еще какое-нибудь приложение для Windows 8, его оно займет оставшееся (большее по размеру) пространство экрана. В противном случае эта часть экрана останется пустой, но и в этом случае

окно текущего приложения займет узкую область экрана, как показано на рис. 4.3.

Рис. 4.3. Приложение в режиме фиксации

Режим фиксации — это особый режим, предназначенный для того, чтобы пользователь мог быстро просмотреть данные какого-либо приложения во время работы другого приложения. Для этого режима лучше всего подходит вертикально ориентированный элемент управления `ListView`, а не горизонтальный `GridView`. Приложения могут работать подобным образом лишь тогда, когда ширина экрана составляет как минимум 1366 пикселов. При меньшем разрешении экрана, например 1024×768, этот режим недоступен. Режим фиксации — это полноценный режим просмотра приложения, хотя область просмотра ограничена значением 320 пикселов в ширину. Этого вполне достаточно для решения большого числа задач, а для смартфонов подобная ширина экрана вполне обычна.

Помимо места, отведенного для окна приложения, в режиме фиксации на экране имеется разделитель размером 22 пиксела. Пользователь может с его помощью перетаскивать окно приложения по экрану. В итоге свободной может остаться область шириной 1024 пиксела, и приложение, работавшее в режиме заполнения, получает в свое распоряжение область размером 1024×768 пикселов (1366 – 320 – 22 = 1024). В случае приложений для Windows 8 это минимально рекомендованное разрешение.

Как вы можете заметить, внешний вид приложения при переходе в режим фиксации меняется. Вместо горизонтально ориентированной сетки, которая содержит элементы, распределенные по строкам и столбцам, на

странице остается вертикальный список. В узкой области просмотра это упрощает навигацию по списку и выбор элементов. Однако гораздо интереснее то, как программа реализует подобное изменение. XAML-разметка и программный код, встроенные в шаблон Grid Application (табличное приложение), используют для этого мощный класс, который называется диспетчером визуальных состояний.

Диспетчер визуальных состояний

Диспетчер визуальных состояний (Visual State Manager, VSM) позволяет отделить управление внешним видом и особенностями поведения пользовательского интерфейса приложения для Windows 8 от его программной логики. Он работает совместно с системой привязки данных, отделяя все, что связано с пользовательским интерфейсом, от остальной логики приложения.

Диспетчер визуальных состояний обрабатывает логику состояний и переходов элементов управления. Понятие «элемент управления» в данном случае не ограничено нестандартными элементами управления или шаблонами элементов управления. Диспетчер одинаково хорошо подходит для управления состоянием страниц и пользовательских элементов управления. VSM работает только с элементами, которые являются наследниками класса `Control`. Среди них страницы (элементы класса `Page`) и пользовательские элементы управления (`UserControl`). Диспетчер визуальных состояний всегда должен представлять собой корневой элемент шаблона элемента управления. В проекте Windows8Application вы обнаружите элемент `VisualStateManager.VisualStateGroups`, который вложен в главную сетку приложения в файле GroupedItemsPage.xaml.

VSM позволяет описывать группы, состояния и, при необходимости, переходы. Группа — это набор взаимоисключающих состояний. Как можно судить по названию, группы позволяют создавать связи между родственными состояниями. В примере, который мы рассматриваем, страница приложения может пребывать в одном из обычных состояний, относящихся к группе `ApplicationViewStates`. Вот эти состояния:

- `FullScreenLandscape` (полноэкранный альбомный режим). Приложение исполняется в полноэкранном режиме, планшетный компьютер находится в альбомной ориентации.
- `Filled` (режим заполнения). Приложение занимает большую часть экрана, в то время как меньшая его часть занята приложением, которое исполняется в режиме фиксации.

- `FullScreenPortrait` (полноэкранный портретный режим). Приложение исполняется в полноэкранном режиме, планшетный компьютер находится в портретной ориентации.
- `Snapped` (режим фиксации). Приложению, которое исполняется в режиме фиксации, выделяется область с краю экрана шириной в 320 пикселов.

Важно понимать, что эти состояния являются взаимоисключающими, так как VSM позволяет странице в некоторый момент времени пребывать лишь в одном из них. Группы отражают контракт, касающийся особенностей режимов работы страницы. Вы задаете набор состояний, в которых может пребывать страница, и поддерживаете функциональность страницы в этих состояниях. Управление состояниями осуществляется с помощью базового класса, от которого унаследована страница. VSM позволяет настраивать визуальные эффекты, происходящие при переходе страницы в то или иное состояние.

Для того чтобы увидеть, как осуществляется управление состояниями, откройте папку `Common` и взгляните на класс `LayoutAwarePage`. Этот класс представляет собой часть шаблона проекта, он применяется при обработке события `ViewStateChanged`, которое вызывается всякий раз, когда меняется ориентация экрана устройства либо происходит переключение между режимами просмотра в пределах неизменной ориентации экрана. Обработчик события получает строковое представление нового состояния просмотра и указывает VSM на то, что нужно перейти в это состояние:

```
VisualStateManager.GoToState(layoutAwareControl, visualState, false);
```

Обратите внимание, в смене состояния просмотра никак не задействована логика пользовательского интерфейса. Это и есть отделение логики приложения от логики пользовательского интерфейса. Приложение занимается управлением состояниями, а изменения в пользовательском интерфейсе, которые соответствуют различным состояниям, описаны в XAML-коде с помощью раскадровок. Если вы задействуете несколько логических групп состояний в применении к одному элементу, это означает, что для данного элемента одновременно работают несколько раскадровок.

Важное правило для групп заключается в том, что они не должны зависеть друг от друга. Хотя элемент управления может существовать одновременно в нескольких состояниях (точнее, к нему может быть применено одно состояние из каждой группы), визуальные эффекты, заданные в разных группах, не должны накладываться друг на друга. Системные средства не обеспечивают выполнение этого правила, так как в каждом из состояний

вы можете описать любые раскадровки. Однако если не соблюдать это правило, результаты могут оказаться совершенно непредсказуемыми.

Группы — это контейнеры для связанных взаимоисключающих состояний. А что такое *состояние*? У этого понятия есть как логическое, так и физическое толкование. Логически оно описывает взаимоисключающий статус элемента. Физически состояние отражает набор визуальных элементов и параметров. Очень важна возможность отделить детали того, как внешне выглядит элемент, пребывающий в некотором состоянии, от самого этого состояния. В программном коде вы можете просто установить элемент управления в некоторое состояние и позволить VSM выполнить перевод элемента в это состояние. Это упрощает не только создание и тестирование элементов управления, но и их расширение и настройку.

Взгляните на страницу GroupedItemsPage.xaml. Здесь объявлена одна безымянная группа, а первое объявленное в ней состояние выглядит так:

```
<VisualState x:Name="FullScreenLandscape"/>
```

Описание состояния не содержит других сведений. Это значит, что существующего XAML-описания страницы достаточно, чтобы визуализировать пользовательский интерфейс для данного состояния. Если вы взглянете на XAML-код, то увидите элемент управления `SemanticZoom`, который объявлен в первой строке сетки (больше о семантическом масштабировании вы узнаете далее в этой главе). В той же строке есть элемент управления `ScrollViewer`, но его свойство `Visibility` установлено в значение `Collapsed`, то есть он, по умолчанию, не визуализируется. Он содержит элемент управления `ListView`, который применяется в режиме фиксации.

Взгляните теперь на описание состояния `Snapped`. Оно приведено в листинге 4.1. Здесь имеется один элемент управления `Storyboard`, содержащий несколько анимаций. Анимации для перемещения элемента не используются. Они служат для модификации значений зависимых свойств. Диспетчер визуальных состояний с помощью анимаций задает значения свойств элементов управления (скоро мы узнаем, почему).

Листинг 4.1. Описание состояния фиксации

```
<VisualState x:Name="Snapped">
                <Storyboard>
                   <ObjectAnimationUsingKeyFrames
  Storyboard.TargetName="backButton"
  Storyboard.TargetProperty="Style">
                    <DiscreteObjectKeyFrame KeyTime="0"
```

```
  Value="{StaticResource SnappedBackButtonStyle}"/>
                </ObjectAnimationUsingKeyFrames>
                <ObjectAnimationUsingKeyFrames
  Storyboard.TargetName="pageTitle"
  Storyboard.TargetProperty="Style">
                    <DiscreteObjectKeyFrame KeyTime="0"
  Value="{StaticResource SnappedPageHeaderTextStyle}"/>
                </ObjectAnimationUsingKeyFrames>

                <ObjectAnimationUsingKeyFrames
  Storyboard.TargetName="itemListScrollViewer"
  Storyboard.TargetProperty="Visibility">
                    <DiscreteObjectKeyFrame KeyTime="0"
  Value="Visible"/>
                </ObjectAnimationUsingKeyFrames>
                <ObjectAnimationUsingKeyFrames
  Storyboard.TargetName="semanticViewer"
  Storyboard.TargetProperty="Visibility">
                    <DiscreteObjectKeyFrame KeyTime="0"
  Value="Collapsed"/>
                </ObjectAnimationUsingKeyFrames>
              </Storyboard>
</VisualState>
```

В этом примере меняются стили кнопки Back (назад) и заголовка. Для соответствующего свойства элемента управления `SemanticZoom` записывается значение `Collapsed`, то есть этот элемент управления и его дочерние элементы в этом режиме на экран не выводятся. У элемента управления `ScrollViewer` свойство `Visibility` установлено в значение `Visible`, в итоге он вместе с содержащимися в нем элементами в данном режиме выводится на экран. Если свойство `Visibility` элемента управления установлено в значение `Collapsed`, он не выводится в визуальном дереве элементов, если в значение `Visible` — выводится, причем даже в том случае, если его свойство непрозрачности равно 0 или он имеет прозрачный (`Transparent`) цвет. Анимация для конкретного визуального состояния будет исполняться до тех пор, пока страница выводится в этом состоянии. Когда состояние меняется, раскадровка (`Storyboard`) останавливается, значения свойств сбрасываются в состояния, предлагаемые по умолчанию (или принимают новые значения, определяемые раскадровкой для нового состояния).

Диспетчер визуальных состояний использует элемент управления `Storyboard`, так как он имеет наивысший приоритет при изменении значе-

ний зависимых свойств. Понять особенности управления состояниями — значит, понять особенности работы с элементами управления `Storyboard`. Когда некий элемент управления переходит в конкретное состояние, VSM останавливает исполнение действий, заданных элементами управления `Storyboard` для других состояний в той же самой группе (не забудьте, состояния в группе являются взаимоисключающими), а затем начинает исполнять действия, заданные элементом `Storyboard` для нового целевого состояния.

Обычно исходное состояние элемента указывают при его инициализации, чтобы элемент управления присутствовал в «графе состояний» или в коллекции правильных состояний. Класс `LayoutAwarePage` делает это в методе `StartLayoutUpdates`. Обратите внимание на вызов `GoToState`, который выполняется перед выходом из метода.

Переходы делают диспетчер визуальных состояний гибким, позволяя ему управлять тем, как элементы переходят от состояния к состоянию. Вы можете задать переход, применяемый всякий раз, когда элемент управления переходит к определенному состоянию, или ограничить его применение только ситуацией, когда элемент переходит из некоторого заданного состояния в любое другое. Переходы — это очень мощный механизм.

В случае самого простого перехода значения существующей раскадровки служат для анимации смены состояний на основе времени перехода. Вы задаете это время, а диспетчер визуальных состояний делает все остальное. Кроме того, можно создавать собственные раскадровки для переходов. Диспетчер визуальных состояний остановит любую раскадровку перехода, как только элемент управления перейдет в новое состояние. Но он не остановит раскадровку во время изменения состояния.

Семантическое масштабирование

Элементы, видимые на главном экране приложения, которое мы рассматриваем в качестве примера, организованы в группы. Вы можете заметить несколько списков, которые содержат множество элементов, формирующих десятки групп. Хотя встроенные элементы управления разработаны для поддержки больших объемов данных, пользователю может быть не совсем удобно перемещаться по подобному списку. Для решения проблемы можно прибегнуть к *семантическому масштабированию* (semantic zoom).

Семантическое масштабирование (его еще называют контекстным масштабированием) — это методика, которая дает пользователю возможность

уменьшить масштаб представления списка и осуществлять навигацию по нему на более общем уровне средствами предоставленного вами интерфейса. Это не визуальное масштабирование, так как когда пользователь меняет масштаб вывода списка, фактически меняется реальный интерфейс. Если в примере Windows8Application уменьшить масштаб списка в окне приложения, либо коснувшись двумя пальцами сенсорного экрана и разведя их, либо вращая колесо мыши, удерживая клавишу Ctrl, либо воспользовавшись клавишей –, вы увидите, как макет с сеткой, представляющий группы и содержащиеся в них элементы, превратится в макет, состоящий из плиток с названиями групп.

Новое представление данных применяется, когда вы уменьшаете масштаб вывода на экран. Это режим общего представления сменяется прежним режимом детального просмотра, когда вы увеличиваете масштаб. Окно приложения со списком в режиме общего просмотра показано на рис. 4.4. В этом режиме легко видеть весь список групп. Если вы прикоснетесь к плитке, представляющей любую группу, автоматически произойдет переход в режим детального просмотра, а фокус окажется на элементах выбранной группы.

Рис. 4.4. Режим общего представления

Этот мощный механизм можно задействовать различными способами. Приложение для чтения лент новостей может в режиме общего представления показывать названия лент, давая пользователю возможность просматривать новости конкретной ленты в режиме детального просмотра. Приложение для работы со списками контактных сведений может выводить на экран первые буквы фамилий, позволяя быстро перейти к разделу с данными о нужных респондентах. С концептуальной точки зрения вы просто показываете пользователю два разных варианта представления данных, основываясь на том уровне детализации, переход к которому он инициировал, прибегнув к семантическому масштабированию.

Семантическое масштабирование реализуют с помощью элемента управления `SemanticZoom`. Откройте файл GroupedItemsPage.xaml в примере Windows8Application. Найдите в разметке описание элемента управления `SemanticZoom`. Режим `SemanticZoom.ZoomedInView` представляет собой обычный режим просмотра контента, который применяется, когда пользователь работает с приложением. В данном примере я просто взял шаблон, предлагаемый по умолчанию, добавил элемент управления `SemanticZoom` и переместил элемент управления `GridView` в раздел `ZoomedView`. Когда пользователь выполняет жест уменьшения масштаба, происходит переход на режим просмотра `SemanticZoom.ZoomedOutView`. Для данного режима я создал набор плиток, основываясь на кратком руководстве Microsoft, которое доступно по адресу http://msdn.microsoft.com/ru-ru/library/windows/apps/xaml/hh781234.aspx.

Вы также можете загрузить пример, относящийся к элементу управления `SemanticZoom` из Windows SDK, на странице http://code.msdn.microsoft.com/windowsapps/GroupedGridView-77c59e8e.

Работая над нашим примером, я изменил код разметки, чтобы на экран выводились названия групп. Вы можете видеть код объявления плиток в файле словаря MyStyles.xaml. Также я использовал событие `OnNavigatedTo`, чтобы подключить источник данных для `GridView`. Взгляните на файл вспомогательного кода, который называется GroupedItemsPage.xaml.cs, и вы найдете там следующий метод:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
  this.DefaultViewModel["Groups"] = e.Parameter;
  this.groupGridView.ItemsSource =
    this.groupedItemsViewSource.View.CollectionGroups;
  base.OnNavigatedTo(e);
}
```

Этот код получает набор групп из источника `CollectionViewSource` в XAML и назначает их источником данных для элемента управления `GridView` в `ZoomedOutView`. Это все, что нужно для использования данного элемента управления. Он за счет собственных встроенных механизмов реагирует на сенсорные жесты масштабирования (управлять масштабированием также можно, нажав клавишу Ctrl и либо прокручивая колесо мыши, либо нажимая клавиши + и -), переключаясь между режимами просмотра контента. Когда пользователь касается какого-либо элемента в режиме общего просмотра (`ZoomedOutView`), элемент управления автоматически переключается в режим детализированного просмотра (`ZoomedInView`) и показывает выбранную группу.

В Microsoft подготовили следующие рекомендации по использованию элемента управления `SemanticZoom`:

- Задайте подходящие размеры целей касания для интерактивных элементов (о целях касания мы поговорим в следующем разделе).
- Создайте удобную интуитивно понятную область масштабирования.
- Используйте структуру, свойственную для данного режима просмотра, например:
  - применяйте групповые названия для элементов групповой коллекции;
  - применяйте упорядочение элементов для несгруппированных коллекций;
  - применяйте страницы для представления коллекций документов.
- Ограничьте количество страниц или экранов в режиме общего просмотра до трех, чтобы пользователь смог быстро переходить к нужным данным.
- Убедитесь, что направление прокрутки контента в различных режимах масштабирования не меняется.
- Не используйте масштабирование для вывода разных наборов элементов (например, показывая один набор элементов в режиме детального просмотра, а другой — в режиме общего просмотра).
- Не задавайте границу для дочерних элементов управления, устанавливайте ее лишь для самого элемента управления `SemanticZoom`.

Полное руководство вы можете найти на странице http://msdn.microsoft.com/ru-ru/library/windows/apps/hh465319.aspx.