

Элемент управления Ribbon

Добавление элемента управления Ribbon

```
<Window x:Class="RibbonTest.MainWindow" ... xmlns:r="clr-namespace:Microsoft.Windows.Controls.Ribbon;assembly=RibbonControlsLibrary">
```

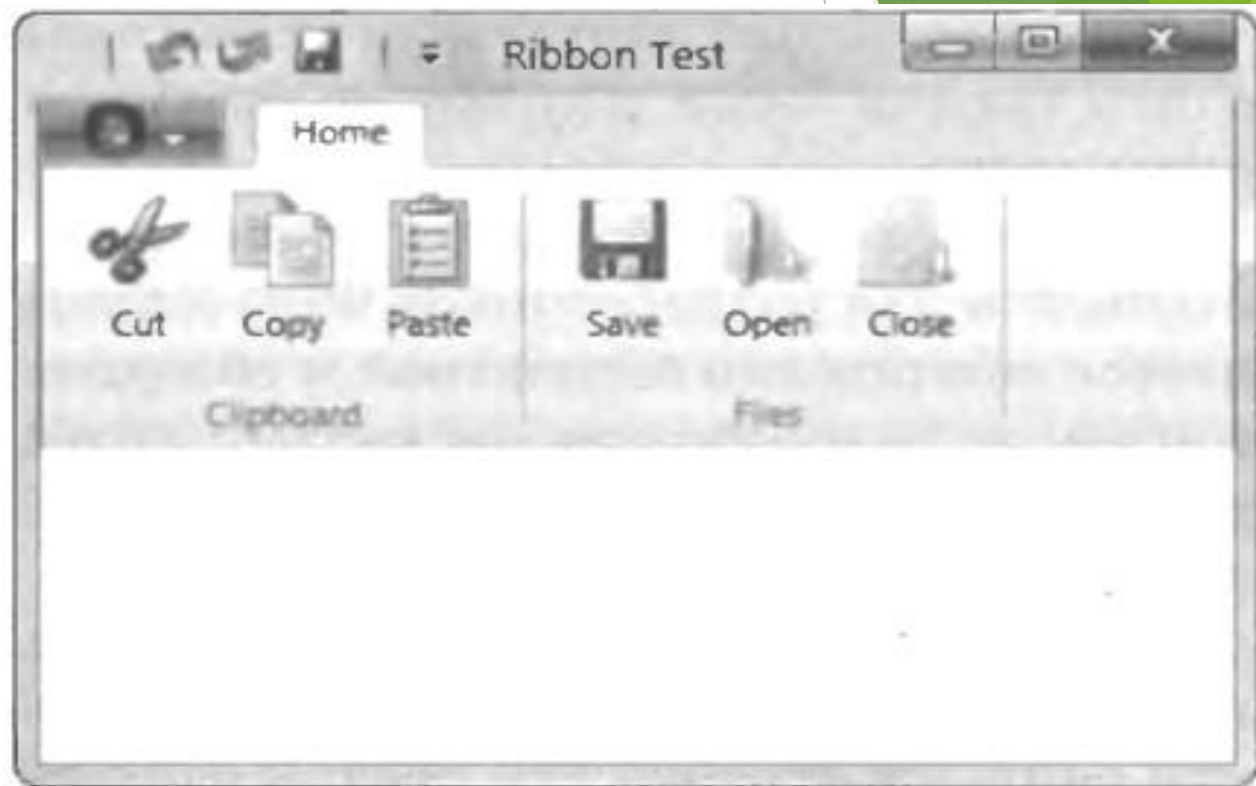
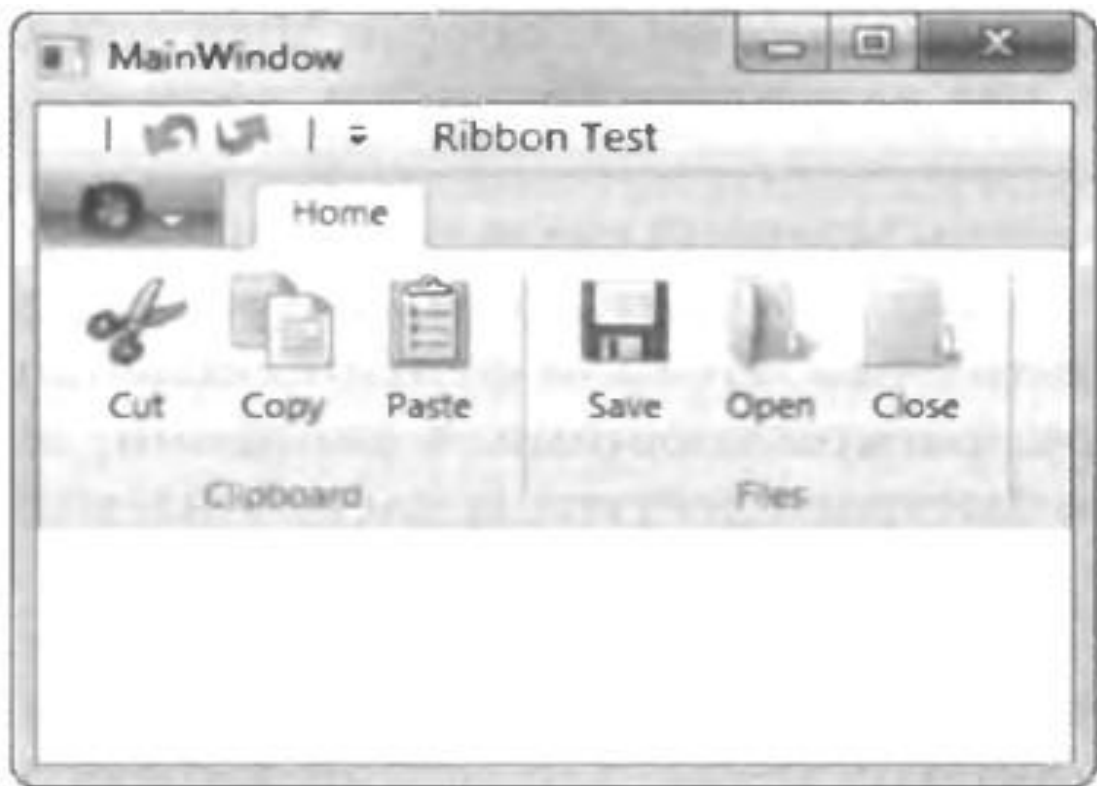


Рис. 25.8. Обычное окно (слева) и окно `RibbonWindow` (справа)

```
<r:RibbonWindow x:Class="RibbonTest.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="350" Width="525"
  xmlns:r=
"clr-namespace:Microsoft.Windows.Controls.Ribbon;assembly=RibbonControlsLibrary">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"></RowDefinition>
      <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>

    <r:Ribbon>
  </r:Ribbon>
  </Grid>
</r:RibbonWindow>
```

```
public partial class MainWindow
{ ... }
```

- ▶ Элемент управления Ribbon состоит из трех фрагментов:
- ▶ панели для быстрого запуска (размещенной сверху),
- ▶ меню приложения (доступное через кнопку, отображаемую в самом конце слева, перед всеми вкладками),
- ▶ ленты с множеством вкладок.

Стилизация элемента управления Ribbon



Рис. 25.9. Элемент управления Ribbon со стилем Windows 7 (слева) и стилем Office 2007 (справа)

- ▶ Готовые коллекции стилей, которые хранятся в сборке RibbonControlsLibrary.dll в виде словаря ресурсов и предоставляются через класс PopularApplicationSkins. Называются они Office2007Blue, Office2007Black и Office2007Silver.

```
public MainWindow()  
{  
    this.Resources.MergedDictionaries.Add(PopularApplicationSkins.Office2007Black);  
    InitializeComponent();  
}
```

Команды

- ▶ В отличие от стандартных меню и панелей инструментов в WPF, в Ribbon перехватывать события Click, поступающие от входящих в его состав элементов управления, не разрешено. Преимущество такого проектного решения в том, что он позволяет поставлять в элементе управления Ribbon более развитую модель команд. (Как рассказывалось в главе 9, базовая модель команд в WPF является довольно скромной.) Недостаток решения в том, что оно не позволяет использовать специальные классы, унаследованные от RoutedCommand.

Меню приложения

```
<r:Ribbon Title="Ribbon Test">  
  <r:Ribbon.ApplicationMenu>  
    <r:RibbonApplicationMenu>  
  
      <r:RibbonApplicationMenuItem>...</r:RibbonApplicationMenuItem>  
      <r:RibbonApplicationMenuItem>...</r:RibbonApplicationMenuItem>  
      <r:RibbonApplicationMenuItem>...</r:RibbonApplicationMenuItem>  
  
    </r:RibbonApplicationMenu>  
  </r:Ribbon.ApplicationMenu>  
</r:Ribbon>
```

```
<r:RibbonApplicationMenuItem>  
  <r:RibbonApplicationMenuItem.Command>  
    <r:RibbonCommand LabelTitle="_Close" LargeImageSource="images\close.png"  
      Executed="Close_Executed" />  
  </r:RibbonApplicationMenuItem.Command>  
</r:RibbonApplicationMenuItem>
```

- ▶ Объект `RibbonApplicationMenu` наивысшего уровня тоже нуждается в объекте `RibbonCommand`, хотя и не используется для запуска команды! Причиной являются несколько других свойств, таких как свойства, связанные с подсказкой, и свойства изображения (которые устанавливают изображение, появляющееся внутри кнопки меню приложения). В случае применения используемого по умолчанию стиля Windows 7 потребуется установить свойство `Small Image`, а для стилей Office 2007, которые предусматривают отображение большой кнопки приложения — свойство `LargeImageSource`.

- ▶ Также важно отметить, что любой `RibbonApplicationMenuItem` может хранить больше объектов `RibbonApplicationMenuItem` для создания подменю, которое отображается во втором столбце меню, как показано на рис. 25.10.

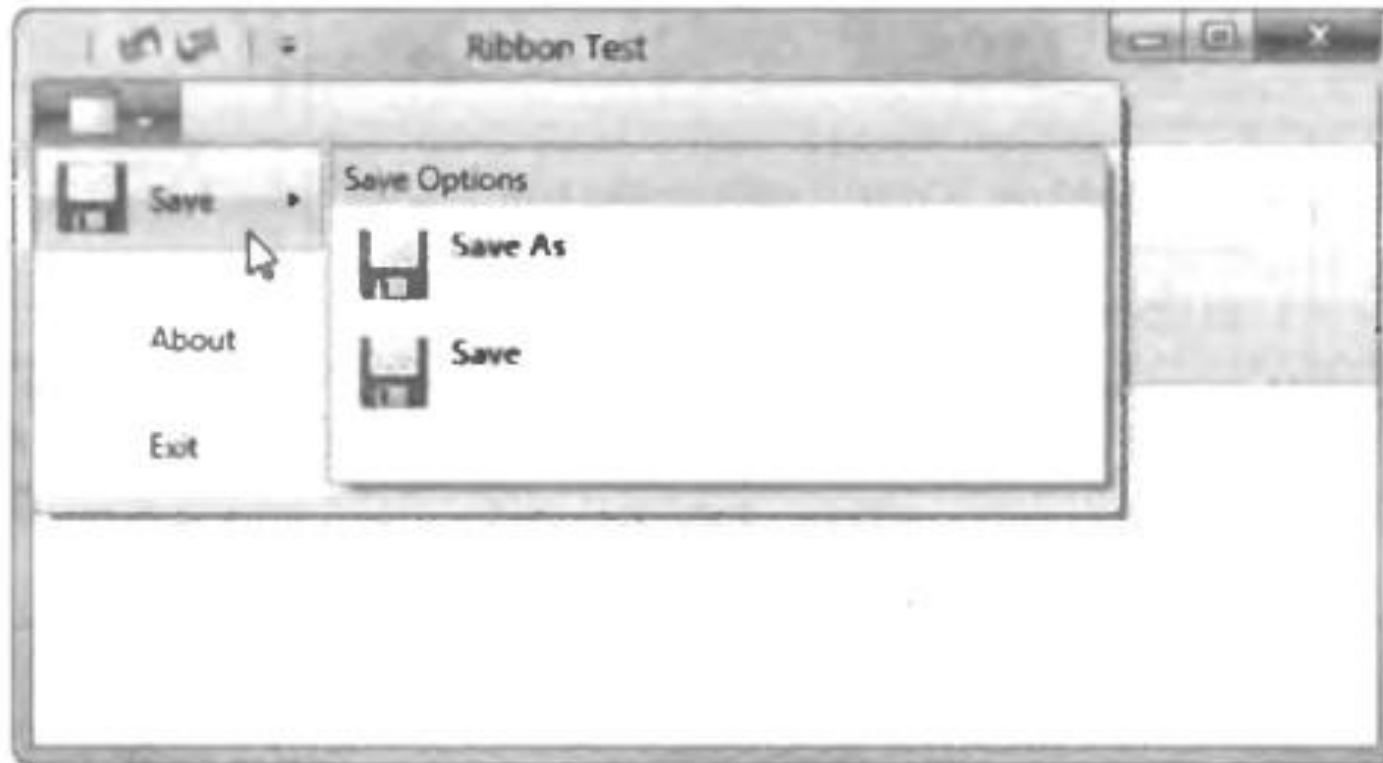
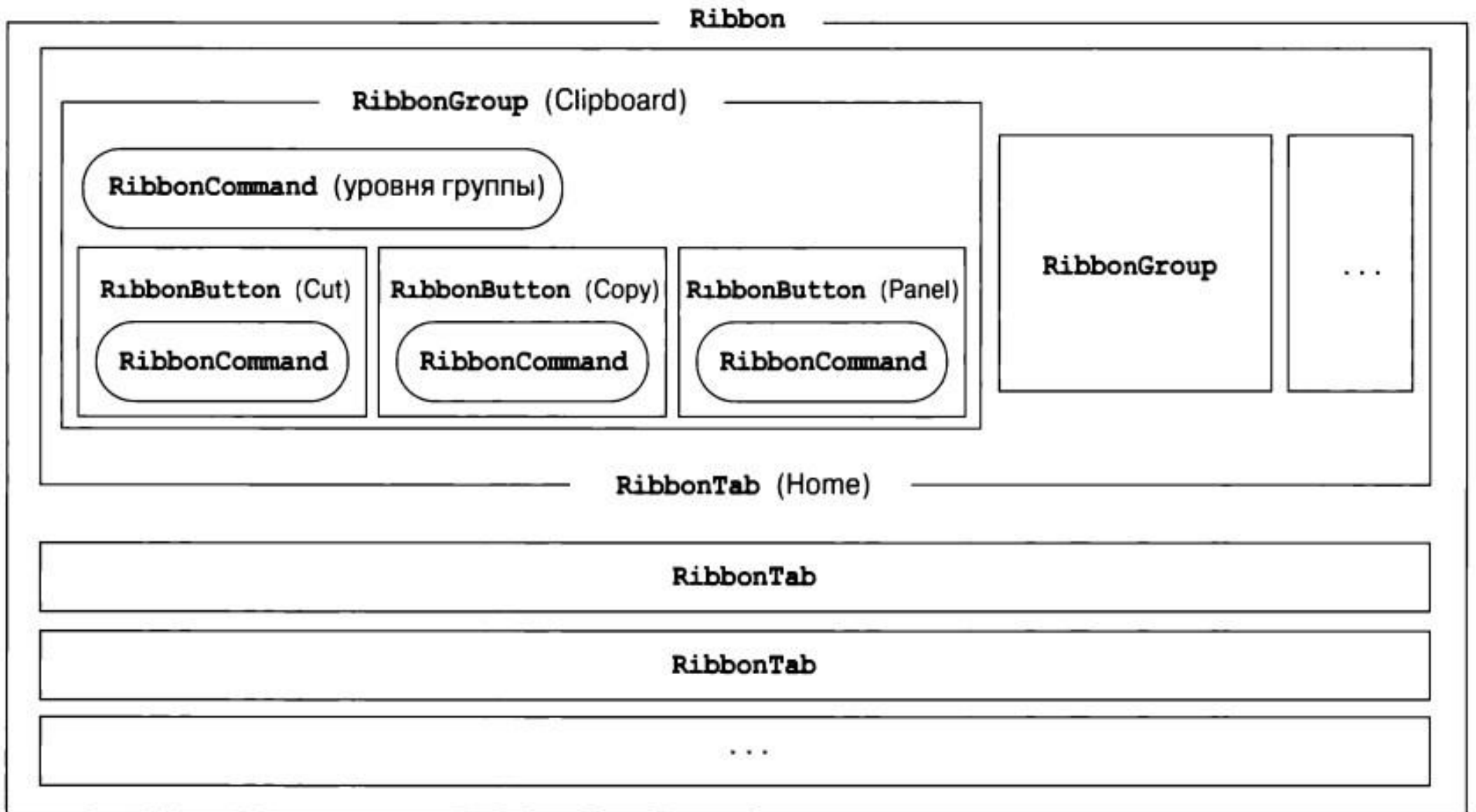


Рис. 25.10. Элемент управления `Ribbon` с подменю

Вкладки, группы и иконки



Потребуется присоединить объект `RibbonCommand` к каждой группе. Этот объект имеет несколько специальных предназначений.

Во-первых, свойство `RibbonCommand.LabelTitle` позволяет указать заголовок группы, который должен отображаться прямо под разделом соответствующей группы в элементе управления `Ribbon`.

Во-вторых, с помощью свойства `RibbonCommand.SmallImageSource` задается изображение, которое должно использоваться в случае нехватки пространства и сворачивания группы в одну кнопку.

И, в-третьих, событие `RibbonCommand.Executed` позволяет создать модуль запуска диалогового окна. (Под модулем запуска диалогового окна подразумевается небольшой значок, который появляется в правом нижнем углу некоторых групп и который при выполнении на нем щелчка приводит к отображению диалогового окна с дополнительными опциями.)

Изменение размеров элемента управления Ribbon



Рис. 25.12. Сжатие элемента управления Ribbon

- ▶ С помощью свойства `RibbonTab.GroupSizeReductionOrder` можно указать, какие группы должны сокращаться первыми, используя для обозначения каждой группы значение ее свойства `LabelText`. Например:

```
<r:RibbonTab Label="Home" GroupSizeReductionOrder="Clipboard, Tasks, File">
```


- ▶ Более мощный подход предусматривает создание коллекции объектов `RibbonGroupSizeDefinition`, которая диктует, каким именно образом должно происходить сворачивание группы.

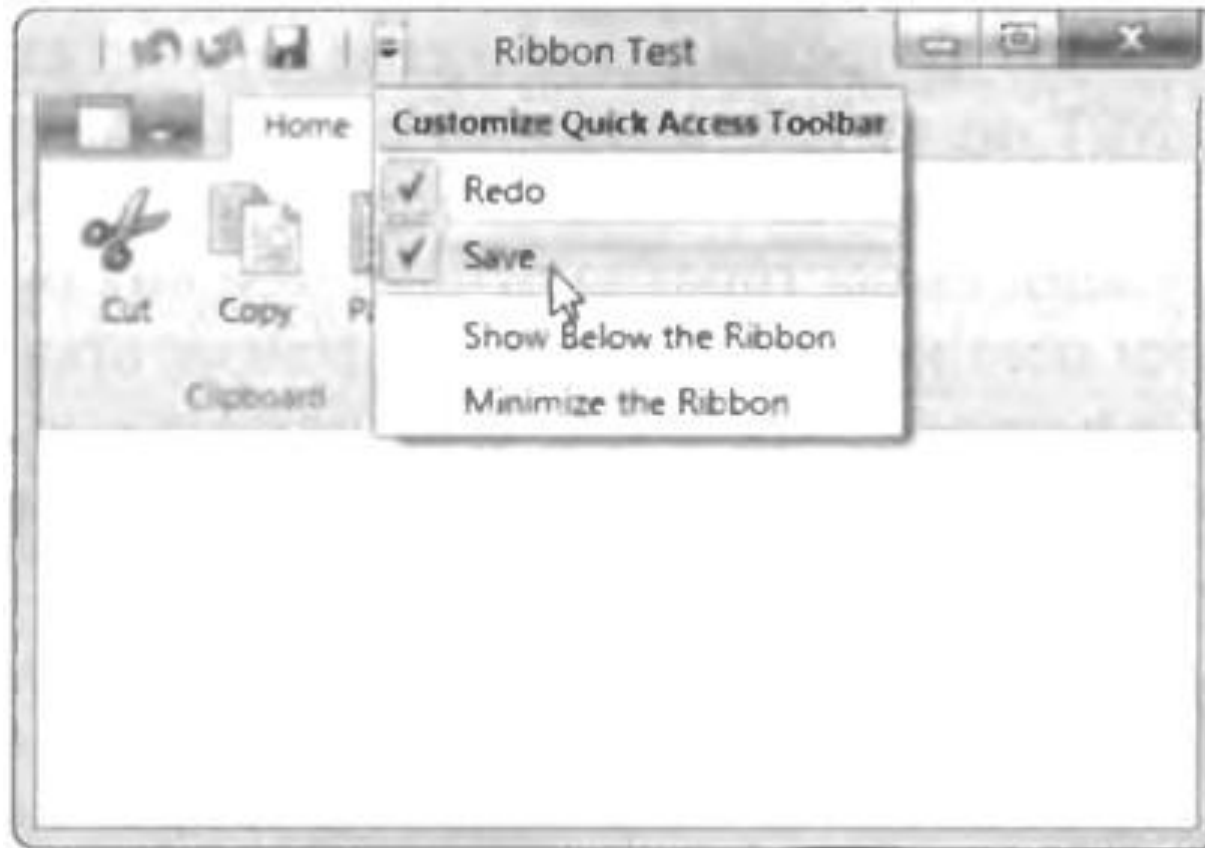
```
<r:RibbonGroupSizeDefinition>  
  <r:RibbonControlSizeDefinition ImageSize="Large" IsLabelVisible="True" />  
  <r:RibbonControlSizeDefinition ImageSize="Large" IsLabelVisible="True" />  
  <r:RibbonControlSizeDefinition ImageSize="Large" IsLabelVisible="True" />  
  <r:RibbonControlSizeDefinition ImageSize="Large" IsLabelVisible="True" />  
</r:RibbonGroupSizeDefinition>
```

- ▶ Чтобы обрести контроль над процессом изменения размеров групп, необходимо определить множество таких объектов `RibbonGroupSizeDefinition` и упорядочить их в коллекции `RibbonGroupSizeDefinitionCollection` от наибольшего к наименьшему.
- ▶ Саму коллекцию `RibbonGroupSizeDefinitionCollection` обычно лучше размещать в разделе `Ribbon.Resources`.

Панель быстрого запуска

- ▶ Панель быстрого запуска представляет объект `QuickAccessToolBar`, который может содержать в себе набор объектов `RibbonButton`. При определении `RibbonCommand` для этих объектов должен быть предоставлен только текст подсказки и небольшое изображение, потому что текстовые метки (`TextLabel`) и крупные изображения в них никогда не отображаются.

- ▶ Единственной новой деталью в панели быстрого запуска является меню настройки, которое появляется в результате щелчка на стрелке раскрывающегося списка в крайней справа ее части. Это меню можно использовать для настройки пользователями команд, появляющихся в панели быстрого запуска. Его можно вообще отключить, установив свойство `QuickAccessToolBar.CanUserCustomize` в `false`.



- ▶ Пользовательская настройка работает через присоединенное свойство `RibbonQuickAccessToolBar.Placement`. Здесь доступны три варианта:
- ▶ Указывайте значение `InToolBar`, если необходимо, чтобы команда появлялась только в панели быстрого запуска (но не в меню настройки), так что она всегда остается видимой.
- ▶ Используйте значение `InCustomizeMenuAndToolBar`, когда нужно, чтобы команда появлялась и в панели быстрого запуска, и в меню, настройки, тогда у пользователя будет возможность снимать с нее отметку и скрывать ее.
- ▶ Применяйте значение `InCustomizeMenu`, если требуется, чтобы команда появлялась в неотмеченном виде в меню настройки, но не в самом элементе управления `Ribbon`, тогда пользователь может ее при необходимости отобразить.

```
<r:Ribbon.QuickAccessToolBar>
  <r:RibbonQuickAccessToolBar CanUserCustomize="True">
    <!-- Всегда видна и не может удаляться. -->
    <r:RibbonButton Command="{StaticResource UndoCommand}"
      r:RibbonQuickAccessToolBar.Placement="InToolBar" />
    <!-- Видна, но может скрываться с помощью меню настройки. -->
    <r:RibbonButton Command="{StaticResource RedoCommand}"
      r:RibbonQuickAccessToolBar.Placement="InCustomizeMenuAndToolBar" />
    <!-- Не видна, но может отображаться с помощью меню настройки. -->
    <r:RibbonButton Command="{StaticResource SaveCommand}"
      r:RibbonQuickAccessToolBar.Placement="InCustomizeMenu" />
  </r:RibbonQuickAccessToolBar>
</r:Ribbon.QuickAccessToolBar>
```