Занятие 9, ч.1. Windows Forms, WPF

Тренер: Алексей Дышлевой

Основные вопросы

- Вступ. Архітектура
- Обробка подій
- Патерни MVC та MVP
- Створення застосування Windows Forms
- Створення користувальницьких елементів управління
- Огляд компонентів Windows Forms
- Основи технології WPF. Архітектура. Фундаментальні класи. Елементи управління. Команди. Ресурси. Стилі. Шаблони.
- Якості. Залежності. Прив’язка даних.
- Основи мови XAML
- Локалізація

Иерархия классов Windows Forms

- System.Object
  System.MarshalByRefObject
  System.ComponentModel.Component
  System.Windows.Forms.Control
  System.Windows.Forms.ScrollableControl
  System.Windows.Forms.ContainerControl
  System.Windows.Forms.Form
- От Form будет наследоваться класс формы окна
- От ContainerControl будут наследоваться пользовательские элементы управления

Класс Control

- Базовый класс элементов управления с визуальным представлением
- Чтобы создать собственный класс элементов управления, осуществите наследование из классов UserControl, Control или из других элементов управления, предоставляемых в Windows Forms.
- Класс Control реализует основные функциональные возможности, необходимые для классов, отображающих сведения для пользователя. Он обрабатывает входные данные, полученные с клавиатуры и указывающих устройств. Он также обрабатывает операции, связанные с маршрутизацией сообщений и безопасностью. Он определяет границы элемента управления (позицию и размер), хотя и не реализует рисование. Данный класс предоставляет дескриптор окна (*hWnd*).

Класс Control

- Элементы управления Windows Forms используют свойства окружения, поэтому дочерние элементы управления могут иметь внешний вид, соответствующий окружающей среде. *Свойство окружения* — это свойство элемента управления, которое (если оно не задано) получается из родительского элемента управления. Если у элемента управления отсутствует свойство Parent и свойство не установлено, элемент пытается определить значение свойства окружения через свойство Site. Если элемент управления не помещен на подложку и если подложка не поддерживает свойства окружения или свойство не задано в объекте AmbientProperties, элемент управления использует собственные значения по умолчанию. Как правило, свойство окружения представляет характеристику элемента управления, такую как свойство BackColor, связанное с дочерним элементом управления. Например, объект Button будет иметь то же свойство BackColor, что и его родительский объектForm по умолчанию. Свойства окружения, предоставляемые классом Control включают: Cursor, Font, BackColor, ForeColor и RightToLeft.

Класс Application

- Предоставляет методы и свойства **static** для управления приложением, например методы для запуска и остановки приложения, для обработки сообщений Windows и свойства для получения сведений о приложении. Этот класс не наследуется
- Класс Application содержит методы для запуска и остановки приложений и потоков, а также для обработки сообщений Windows следующим образом:
- Run запускает цикл обработки сообщений приложения в текущем потоке и при необходимости делает форму видимой.
- Exitили ExitThread останавливает цикл обработки сообщений.
- DoEvents обрабатывает сообщения пока программа в цикле.
- AddMessageFilter добавляет фильтр сообщений к насосу сообщения приложения с сообщениями окна монитора.
- IMessageFilter позволяет остановить событие или выполнять особые операции до вызова обработчика событий.
- Этот класс содержит CurrentCulture и CurrentInputLanguage свойства для получения или задания данных о языке и региональных параметрах для текущего потока.
- Нельзя создать экземпляр этого класса.

Пространство имен System.Windows.Forms

- Пространство имен **System.Windows.Forms** содержит классы для создания приложений Windows, которые позволяют наиболее эффективно использовать расширенные возможности пользовательского интерфейса, доступные в операционной системе Microsoft Windows. Содержит:
- Элементы управления, пользовательские элементы управления и формы. Большинство классов в пространстве имен **System.Windows.Forms** являются производными от класса Control. Класс Controlпредоставляет основные функциональные возможности для всех элементов управления, отображаемых в Form. КлассFormпредставляет окно в приложении. Оно включает диалоговые окна, немодальные окна, а также клиентские и родительские окна интерфейса MDI. Можно также создать собственные элементы управления путем наследования от класса UserControl.

Пространство имен System.Windows.Forms

- Меню и панели инструментов. Windows Forms включает широкий набор классов, которые позволяют создавать пользовательские панели инструментов и меню, отличающиеся современным обликом и поведением. ToolStrip , MenuStrip, ContextMenuStrip и StatusStrip позволяют создавать панели инструментов, строки меню, контекстные меню и строки состояния, соответственно.
- Макет. Несколько важных классов в Windows Forms помогают контролировать расположение элементов управления на отображаемой поверхности, например в форме или элементе управления. FlowLayoutPanel располагает все элементы управления которые содержит в последовательном режиме, а TableLayoutPanel позволяет определять ячейки и строчки для расположения элементов управления в фиксированной сетке. SplitContainer разделяет поверхность отображения на две или более корректируемых части.

Пространство имен System.Windows.Forms

- Элементы управления. Пространство имен **System.Windows.Forms** предоставляет большое количество классов элементов управления, которые позволяют создавать пользовательские интерфейсы с расширенными возможностями. Некоторые элементы управления предназначены для ввода данных в приложении, например элементы TextBox и ComboBox. Другие элементы управления отображают данные приложений, например Label и ListView. Это пространство имен также предоставляет элементы управления для вызова команд в приложении, например Button. Элемент управления WebBrowser и такие классы управляемых HTML-страниц, как HtmlDocument, позволяют отображать HTML-страницы и выполнять с ними определенные действия в области управляемого приложения Windows Forms. Элемент управленияMaskedTextBox представляет собой улучшенный элемент управления вводом данных, который позволяет создавать маску для принятия или отклонения введенных пользователем данных в автоматическом режиме.
  Кроме того, с помощью элемента управления PropertyGrid можно создать собственный конструктор Windows Forms, отображающий видимые конструктором свойства элементов управления.

Пространство имен System.Windows.Forms

- Данные и привязка данных. Windows Forms обеспечивает расширенную архитектуру для привязывания к таким источникам данных, как базы данных и XML-файлы.Элемент управленияDataGridView предоставляет настраиваемую таблицу для отображения данных и позволяет настраивать формат ячеек, строк, столбцов и границ.
  Элемент управленияBindingNavigator представляет стандартный способ навигации и работы с данными в форме; BindingNavigator часто используется в сочетании с элементом управления BindingSource для перемещения от одной записи к другой в форме, а также для выполнения операций с записями.
- Компоненты. Помимо элементов управления пространство имен **System.Windows.Forms** предоставляет другие классы, которые не являются производными от класса Control, но также обеспечивают визуальные функции для приложений Windows. Такие классы, как ToolTip иErrorProvider, расширяют возможности или предоставляют сведения пользователям. Классы Help и HelpProvider позволяют отображать текст справки для пользователя, который работает с приложениями.

Пространство имен System.Windows.Forms

- Общие диалоговые окна. Windows предоставляет несколько основных диалоговых окон, позволяющих обеспечить единообразие пользовательского интерфейса в приложениях Windows при выполнении таких операций как открытие и сохранение файлов, задание цвета шрифта или текста и печать.КлассыOpenFileDialog и SaveFileDialog предоставляют возможность отображения диалогового окна, в котором пользователь может выполнить поиск файла, а также ввести имя файла, который необходимо открыть или сохранить. Класс FontDialog отображает диалоговое окно для изменения элементов Font, используемого приложением. Классы PageSetupDialog, PrintPreviewDialog и PrintDialog отобра жают диалоговые окна, позволяющие пользователю управлять параметрами печати документов. Дополнительные сведения о печати с помощью приложений Windows см. в разделе, посвященном пространству имен System.Drawing.Printing. Помимо основных диалоговых окон пространство имен **System.Windows.Forms** предоставляет класс MessageBox для отображения окна сообщения, в котором могут отображаться и извлекаться данные пользователя.

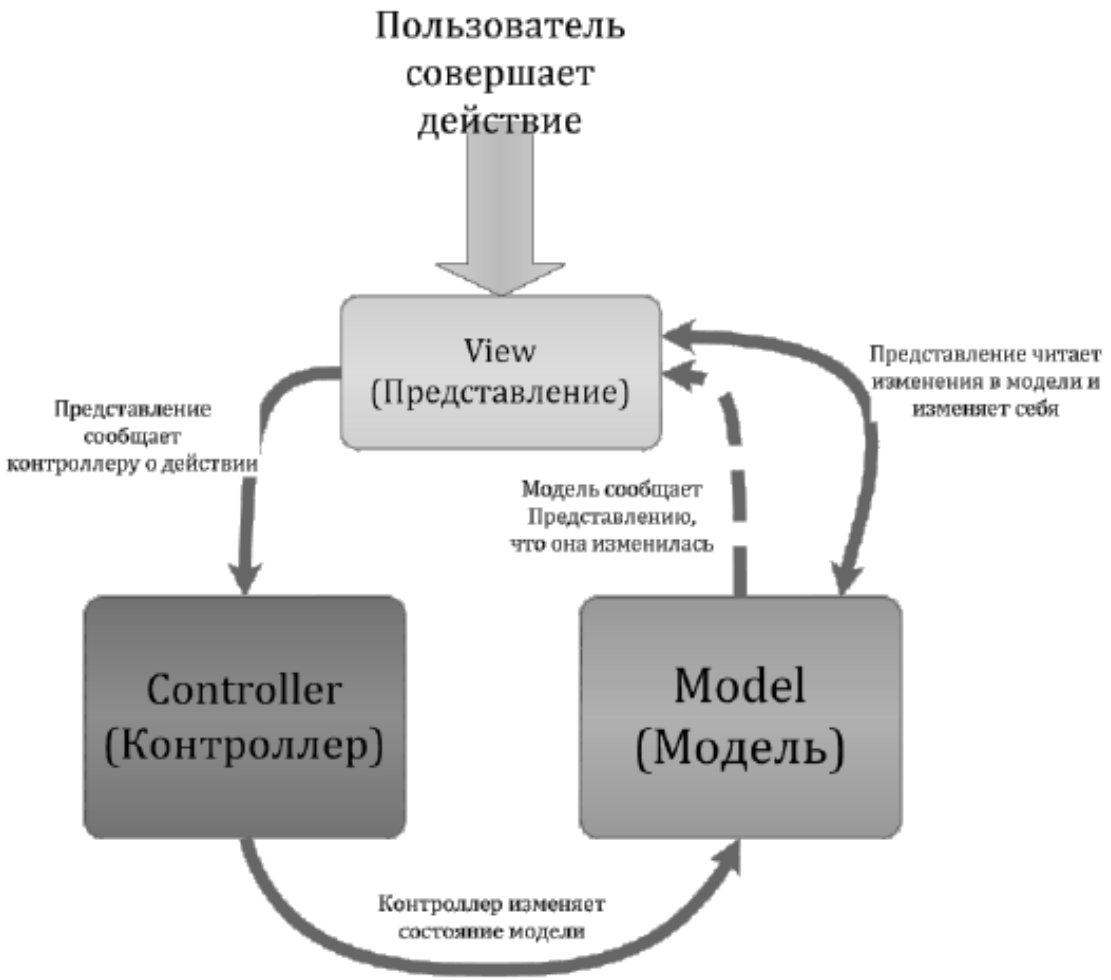
Пример

- Создание простого приложения

MVC

- Шаблон проектирования MVC разделяет работу приложения на отдельные роли:
- Модель данных (model)
- Пользовательский интерфейс (view)
- Управляющую логику (controller)
- Таким образом, изменения, вносимые в один из компонентов, оказывают минимально возможное воздействие на другие компоненты

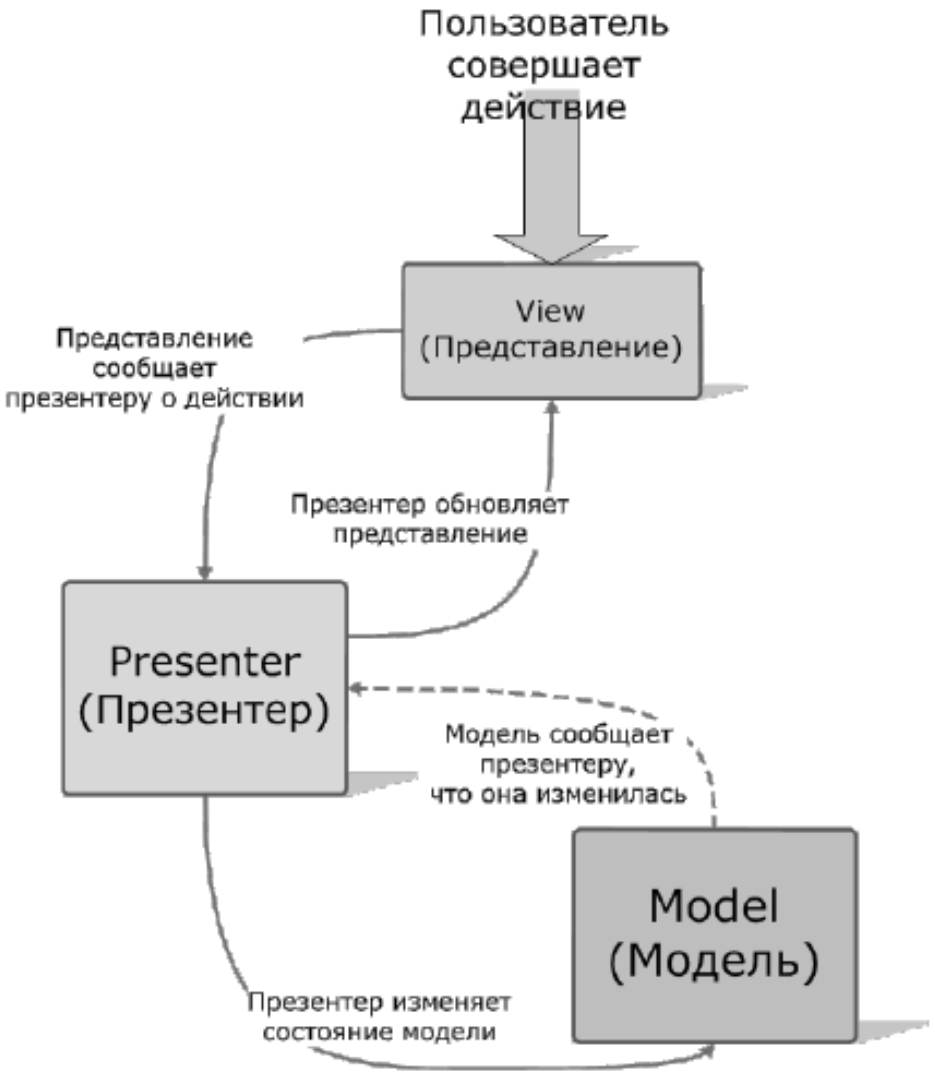
MVC


Пользователь
совершает
действие
View
(Представление)
Представление
сообщает
контроллеру о действии
Представление читает
изменения в модели и
изменяет себя
Модель сообщает
Представлению,
что она изменилась
Controller
(Контроллер)
Model
(Модель)
Контроллер изменяет
состояние модели

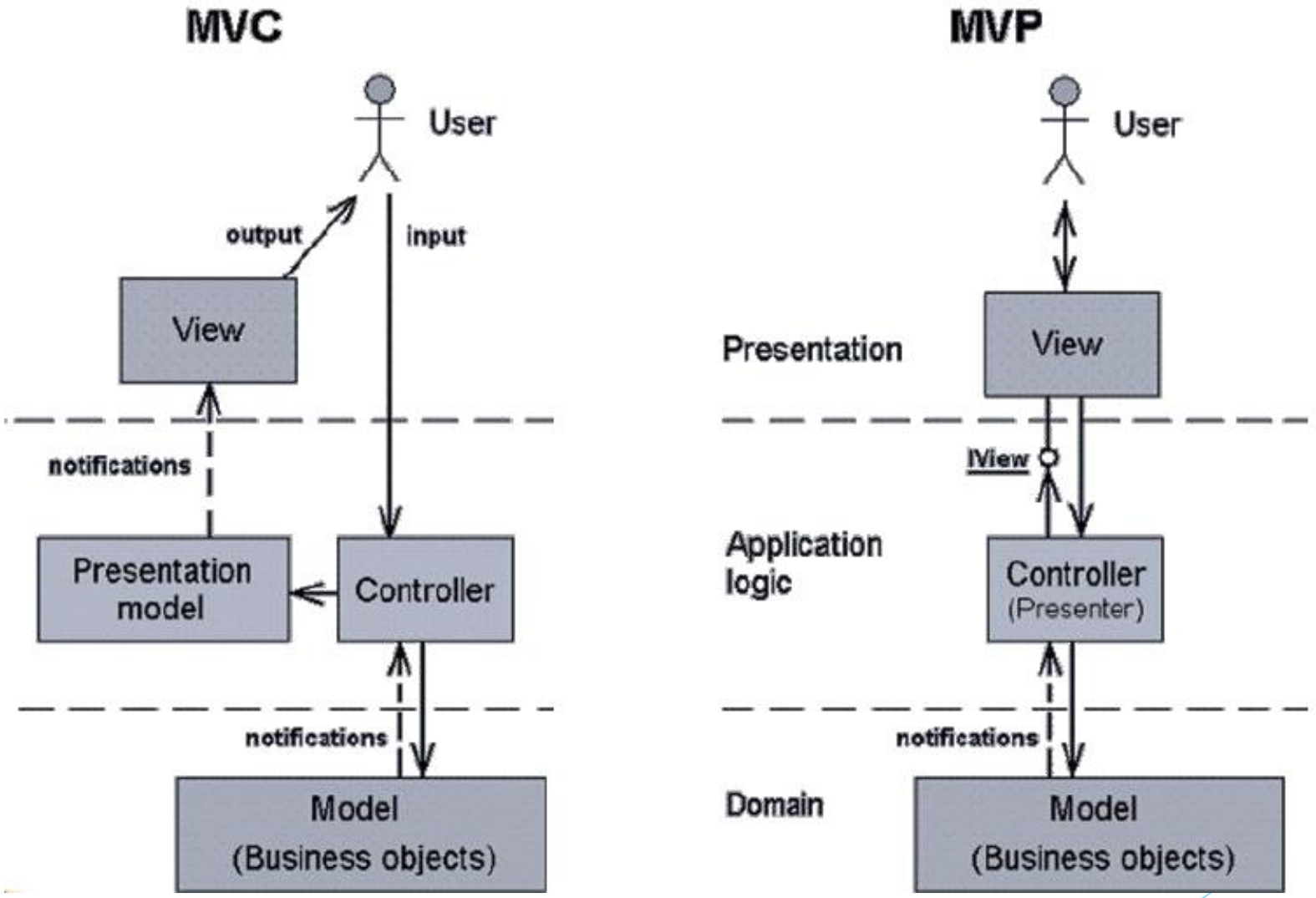
MVP

- Модель (model) представляет собой интерфейс, определяющий данные для отображения или участвующие в пользовательском интерфейсе иным образом
- Вид (view) - это интерфейс, который отображает данные (модель) и маршрутизирует пользовательские команды (или события) для Presenter, чтобы тот действовал над этими данными
- Presetner действует на моделью и видом. Он извлекает данные из хранилища (модели), и форматирует их для отображения в Виде (view). Также реализует обработку событий вида

MVP


Пользователь совершает действие
View (Представление)
Представление сообщает презентеру о действии
Презентер обновляет представление
Presenter (Презентер)
Модель сообщает презентеру, что она изменилась
Model (Модель)
Презентер изменяет состояние модели

MVC vs MVP


MVC
User
output
input
View
notifications
Presentation model
Controller
notifications
Model
(Business objects)
MVP
User
Presentation
View
IView
Application logic
Controller
(Presenter)
notifications
Domain
Model
(Business objects)

Windows Forms + MVP

- Пример 1

Элементы управления Windows Forms

- http://msdn.microsoft.com/ru-ru/library/3xdhey7w(v=vs.110).aspx
- http://msdn.microsoft.com/ru-ru/library/xfak08ea(v=vs.110).aspx

Пользовательские элементы управления

- UserControl это элемент управления, на котором группируются другие элементы и используются все вместе
- http://www.akadia.com/services/dotnet_user_controls.html

Пример

Windows Forms

- Это технология интеллектуальных клиентов для .Net Framework. Представляется набором управляемых библиотек, обеспечивающих распространенные задачи приложений.
- Интеллектуальный клиент – это приложение с богатым графическим интерфейсом, простое в развертывании и обновлении, способное работать при наличии или отсутствии подключения к Интернету и использующее более безопасный доступ к ресурсам на локальном компьютере по сравнению с традиционными приложениями Windows.
- В Windows Forms форма является видимой поверхностью, на которой отображается информация для пользователя. Обычно, приложение Windows Forms строится путем помещения элементов управления на форму и написанием кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш.
- Элемент управления – это отдельный элемент пользовательского интерфейса, предназначенный для отображения и ввода данных.

Особенности WPF

- *Расширенная поддержка для разработки клиентских приложений*. Разработчики могут создавать привлекательные, высоко функциональные приложения. WPF включает в себя отдельные тексто-рендеринговые возможности, такие как **OpenType** и **TrueType**.
- *Простота дизайна пользовательского интерфейса*. WPF предоставляет набор встроенных элементов управления. WPF использует концепцию, согласно которой существует разделение логики элемента управления от его внешнего вида, что считается хорошим архитектурным принципом.
- *Использование XAML*. XAML позволяет разработчикам использовать XAML-модели для декларативного управления объектной моделью. XAML быстрее и проще в реализации, чем процедурный код. XAML используется для определения пользовательского интерфейса в приложениях WPF.
- *Поддержка совместимости со старыми приложениями*. Разработчики могут использовать WPF внутри существующего кода Win32 или существующий код Win32 в WPF

Структура WPF

- Файл с расширением **.csproj** для представления проекта WPF и структурирует в проекте WPF все компоненты по умолчанию.
- Ссылки на необходимые сборки, включая сборки PresentationCore, PresentationFramework, System, System.Core и System.Xaml.
- Файл разметки App.xaml и файл кода (code-behind) App.xaml.cs, которые можно использовать для определения ресурсов и функциональности уровня приложения.
- Файл разметки MainWindow.xaml и файл кода (code-behind) MainWindow.xaml.cs, которые можно использовать в качестве отправной точки для создания первого окна WPF

Использование XAML

- XAML является языком разметки для декларативного программирования приложений. Использование разметки XAML во время разработки позволяет отделить дизайн пользовательского интерфейса от логики приложения, которая хранится в файлах кода. XAML непосредственно представляет экземпляры управляемых объектов

Разметка MainWindow.xaml

- <Window x:Class="WpfApplication1.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">
- <Grid>
- </Grid>
- </Window>
- Эта разметка определяет простое окно с названием, шириной и высотой по умолчанию. Изменить эти свойства можно, редактируя код XAML, или с помощью окна свойств в Visual Studio. Можно также изменить эти свойства динамически, с помощью кода при запуске приложения.
- Элемент управления (control) Grid регулирует расположение элементов управления, которые добавляются к окну. Если нужно использовать альтернативное расположение, можно заменить разметку для элемента управления Grid другим элеметном управления.

Разметка App.xaml

- <Application x:Class="WpfApplication1.App"
- xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
- xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
- StartupUri="MainWindow.xaml">
- <Application.Resources>
-
- </Application.Resources>
- </Application>
- Элемент Application содержит атрибут **StartupUri**, который указывает на окно, которое будет открываться при запуске приложения.

Библиотека элементов управления WPF

Элемент управления	Описание	Пример XAML кода
Button	Представляет собой типичную интерактивную (clickable) кнопку, которую можно найти в большинстве Windows приложений.	<Button Name="myButton" BorderBrush="Black" BorderThickness="1" Click="myButtonOnClick" ClickMode="Press">Click Me</Button>
Canvas	Определяет область, в рамках которой можно явно расположить дочерние элементы с помощью использования координат, являющихся относительными к области Canvas.	<Canvas Background="Black" Height="200" Width="200"> <!-- Child controls --> </Canvas>
ComboBox	Представляет собой раскрывающийся список, прокрутив который, пользователь может сделать выбор.	<ComboBox Name="myComboBox" SelectionChanged= "myComboBox_SelectionChanged"> <ComboBoxItem>Item a</ComboBoxItem> <ComboBoxItem>Item b</ComboBoxItem> </ComboBox>
Grid	Представляет собой гибкую таблицу, которая может содержать несколько столбцов и строк. Как правило, элемент управления Grid используется для расположения дочерних элементов управления.	<Grid ShowGridLines="True" Width="200" Height="200"> <Grid.ColumnDefinitions> <ColumnDefinition/> <ColumnDefinition/> </Grid.ColumnDefinitions> <Grid.RowDefinitions> <RowDefinition /> </Grid.RowDefinitions> <!-- Child controls --> </Grid>
Label	Представляет текстовый блок только для чтения, который можно использовать для отображения некоторого статичного текста.	<Label Name="myLabel">Hello</Label>
StackPanel	Позволяет стековать дочерние элементы управления по горизонтали или по вертикали.	<StackPanel Name="myStackPanel" Orientation="Vertical"> <Label>Item 1</Label> <Label>Item 2</Label> <Label>Item 3</Label> </StackPanel>
TextBox	Представляет редактируемые поля, которые можно использовать для отображения и охватывания текста.	<TextBox Name="myTextBox"></TextBox>

Элементы управления WPF

- Можно определить элементы управления динамически с помощью Visual C# в файле кода.
- Каждый элемент управления в WPF имеет связанный с ним набор свойств, которые можно использовать для определения внешнего вида и поведения элемента управления. Например, большинство элементов управления имеют свойства Height и Width, которые определяют его размеры, и свойство Margin, указывающее, где элемент управления должен появиться относительно элемета управления макета, в котором он содержится.
- Свойства элементов управления можно установить:
- Декларативно в окне XAML путем редактирования напрямую XAML определения.
- В окне Properties. Такой подход изменяет XAML определение элементов управления.
- Во время выполнения с помощью кода Visual C#. Этот подход не меняет XAML определение элемента управления.

События в приложениях WPF

- При создании WPF, ASP.NET и Windows Forms приложений Visual Studio, создаются приложения, управляемые событиями, т.е. выполняющие код в ответ на события. Любая создаваемая форма и элемент управления предоставляют предопределенный набор событий. Когда происходит одно из этих событий, и существует код, связанный с обработчиком этого события, код вызывается.
- Указать события, на которые реагирует элемент управления, можно во время разработки путем редактирования XAML определения элемента управления (указав событие и имя метода обработки события для запуска при наступлении события). Кроме того, можно использовать вкладку Events в окне Properties (эта техника автоматически изменяет XAML определение элемента управления). Необходимо предоставить методы, обрабатывающие события с помощью кода в файле кода.

Пример

- [XAML control declaration]
- <Button Name="myButton" Click="myButton_Click">ClickMe</Button>
-
- [Visual C# event handler]
- private void myButton_Click(object sender, RoutedEventArgs e)
- {
- // Code to do something goes here.
- }

Пример

- [XAML control declaration]
- <Window x:Class="WpfApplication1.MainWindow" Name="myWindow"
- xmlns="..."
- xmlns:x="..."
- Title="MainWindow" Height="350" Width="525" Closing="myWindow_Closing">
- </Window>
-
- [Visual C# event handler]
- private void myWindow_Closing(object sender,System.ComponentModel.CancelEventArgs e)
- {
- // Code to do something goes here.
- }

Создание приложения WPF

- Создать приложение
- Добавить элемент управления
- Установить свойства элемента управления
- Добавить обработчики событий
- Добавить код в приложение

Приложение WPF

Анимация WPF

- http://msdn.microsoft.com/ru-ru/library/ms752312(v=vs.90).aspx
- http://msdn.microsoft.com/ru-ru/magazine/cc163397.aspx
- http://professorweb.ru/my/WPF/graphics_and_animation/level12/graph_animation_index.php

Архитектурный паттерн MVVM

- Model
- View
- ViewModel

Домашнее задание

- Создать приложение для синхронизации файлов в разных директориях (например, диск и флеш-накопитель).
- Реализовать функции изменения, удаления, создания файла для каждой из синхронизируемых директорий
- Использовать архитектурный шаблон MVP для Windows Forms