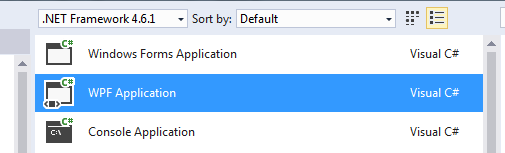
## C# Уровень 3.

# Продвинутый курс с WPF.

# Урок 3

# Создание DLL. Создание своих элементов управления.



Создание собственных библиотек DLL. Создание своих контролов (элементов управления).

# Введение.

Сегодня речь пойдет о создании динамических библиотек и своих собственных элементов управления.

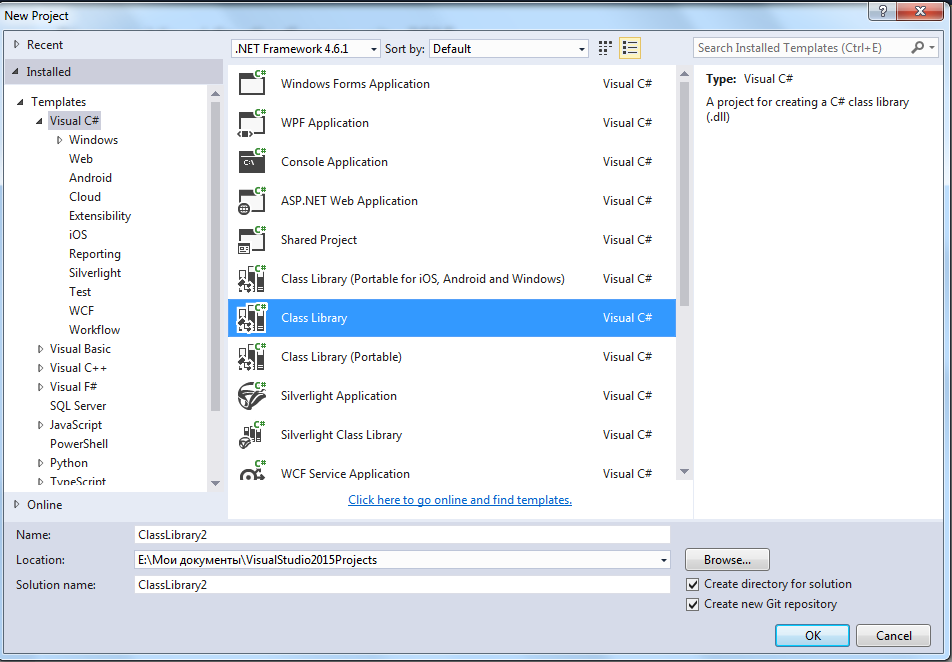
# DLL.

## Что такое DLL.

DLL – это библиотека динамической компоновки, динамическая библиотека или динамически подключаемая библиотека или по английски dynacmiс link library. Служит целью многократного использования различными программными приложениями.

## Создание файла DLL.

Создадим новый проект, выберем тип проекта Class Library.

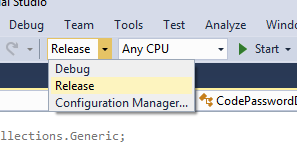


Назевем новый проект CodePasswordDLL.

Класс назовем просто CodePassword. Наименования пространства имен и класса, лучше делать разные, иначе потом будут сложности с использованием этой библиотеки.

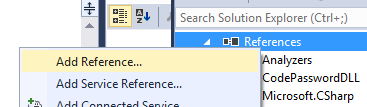
Теперь перенесем код из класса PasswordClass нашего проекта MailSender в новый проект по созданию DLL.

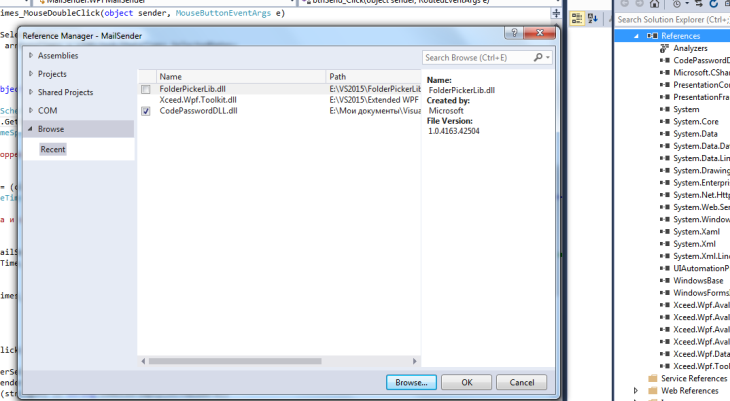
Переключим проект из Debug в Release



И соберем проект. В папке Bin->Release мы найдем CodePasswordDLL.dll файл.

Теперь возвращаемся в наш проект MailSender и подключаем новую библиотеку к ссылкам проекта





Закодированный пароль мы использовали в файле VariableClass.cs

Значит там нужно добавить using CodePasswordDLL;

И немного подправить код.

|  |
| --- |
| using System.Collections.Generic;  using CodePasswordDLL;  namespace MailSender  {  public static class VariablesClass  {  public static Dictionary<string, string> Senders  {  get { return dicSenders; }  }  private static Dictionary<string, string> dicSenders = new Dictionary<string, string>()  {  { "79257443993@yandex.ru", CodePassword.getPassword("{3t1l2m6") },  { "sok74@yandex.ru", CodePassword.getPassword("{3t1l2m6") }  };    }  } |

И удаляем файл PasswordClass.cs из проекта.

Запускаем, всё работает.

В принципе, можно не создовать отдельное решение, а в том же решении добавить ещё один проект.

## Помещаем DLL в GAC.

При создании DLL мы создали закрытую сборку. Использовать закрытую сборку, можно просто подключив к любому проекту и положив рядом с файлом .exe.

Если мы хотим использовать сборку не с одним проектом, а с несоколькими, то и положить надо её во всех этих проектах.

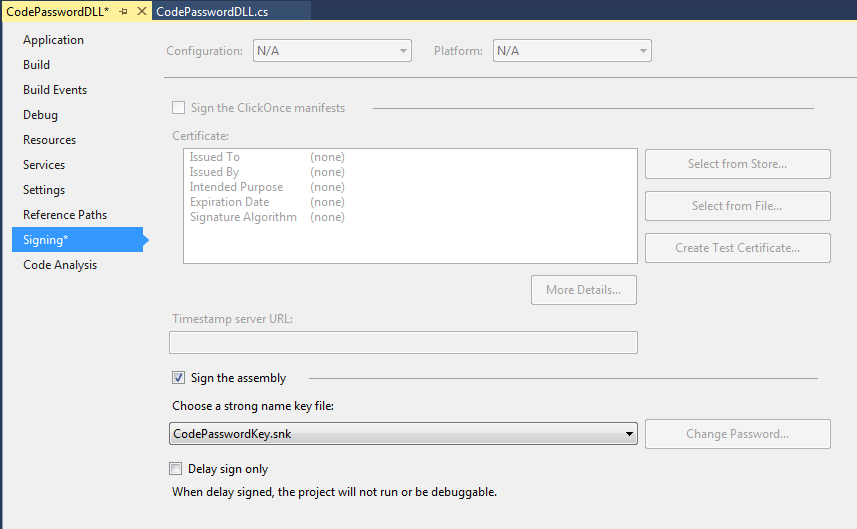
Но мы можем положить DLL-ку в GAC или Global Assemble Cache, глобальный кэш сборок.

Что бы положить dll-ку в кэш, ей нужно задать строгое имя, в которое входит

1. Наименование сборки без расширения
2. Версия. При помощи номера версии мы можем воспользоваться разными версиями одной и той же сборки
3. Открытый ключ
4. Цифровая подпись

Кликаем правой кнопкой мыши по проекту, который создает DLL.

Нажимаем Properties и идем в Signing



Выбираем галочку Sing the assembly, далее в списке Choose a strone name key file, выбираем New и придумываем имя ключа, например CodePasswordKey.snk

Снова запускаем Release

Теперь надо поместить dll в gac.

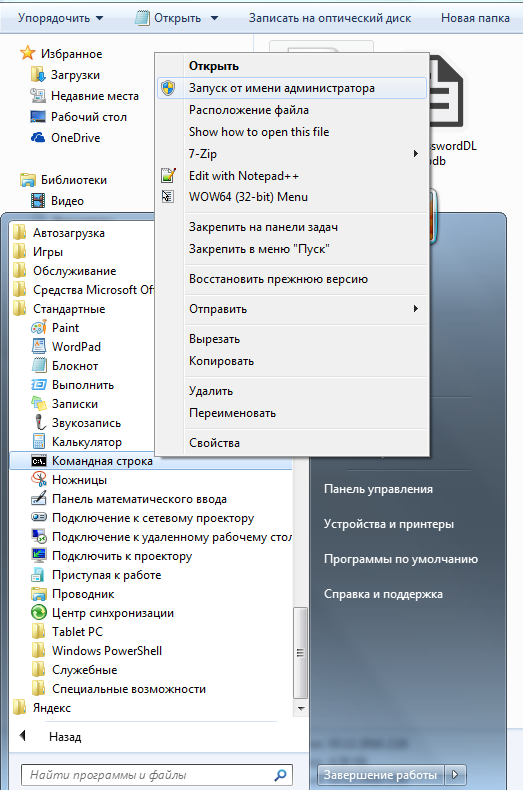
Для этой цели нам понадобится утилита gacutil.exe, у меня на компьютере она расположена здесь

C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools

У этой утилиты есть три команды

1. –i помещаем dll в gac
2. –l просмотреть весь список сборок
3. –u удалить из gac

Заходим в консоль под администратором

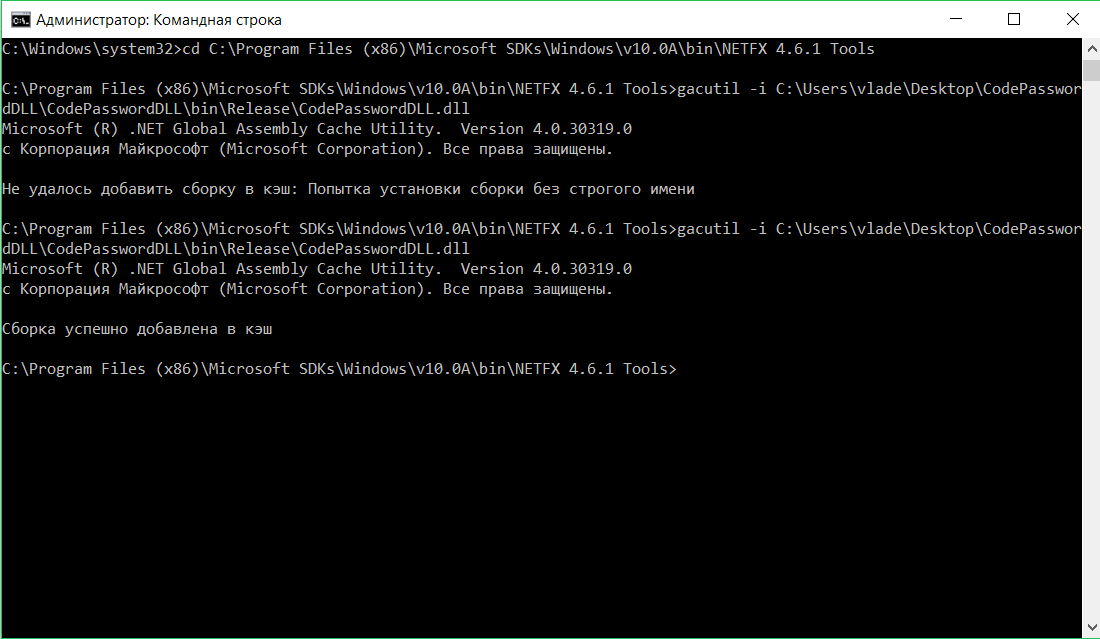


Переходим в расположение утилиты – вводим в командной строке “cd” и путь к папке утилиты

C:\Windows\system32>cd C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools

Затем вводим gacutil -i Полное\_имя\_сборки

C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Tools>gacutil -i C:\Users\vlade\Desktop\CodePasswordDLL\CodePasswordDLL\bin\Release\CodePasswordDLL.dll

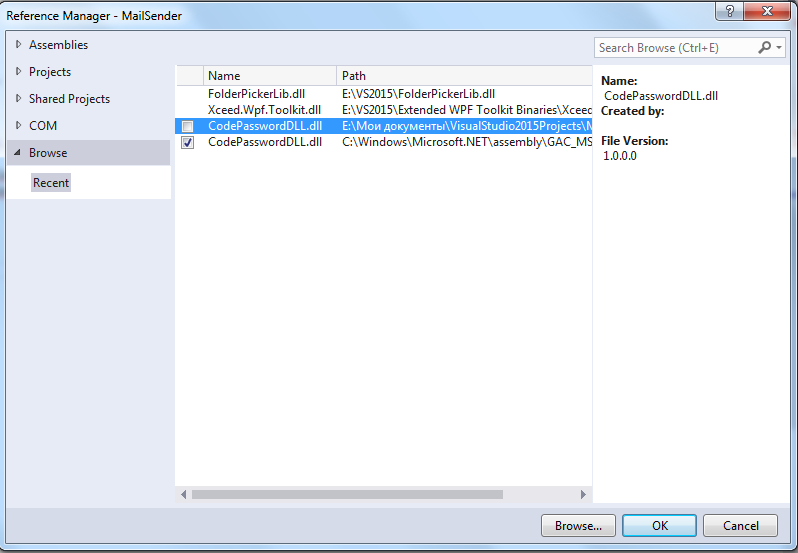


При успешном добавлении dll в gac появится надпись в консоли.

Assembly successfully added to the cache. Если сначала появилась ошибка как у меня, значит не сделали Release

Теперь попробуем подключиться к библеотеке из GAC.

Удаляем сборку CodePasswordDLL.dll из Referencies и добавляем снова, но уже из GAC



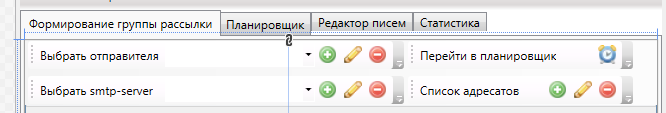
# Создание собственного элемента управления (контрола).

Есть два способа создать свой контрол.

1. Композиция или составной элемент управления. То есть из существующих контролов создается новый контрол. Наследуется от UserControl.
2. Расширение функционала другого контрола. CustomControl

## Создаём составной элемент управления.

Если мы посмотрим на нашу программу «Рассыльщик», на первую вкладку, то увидим там ToolBarTray с панелями.

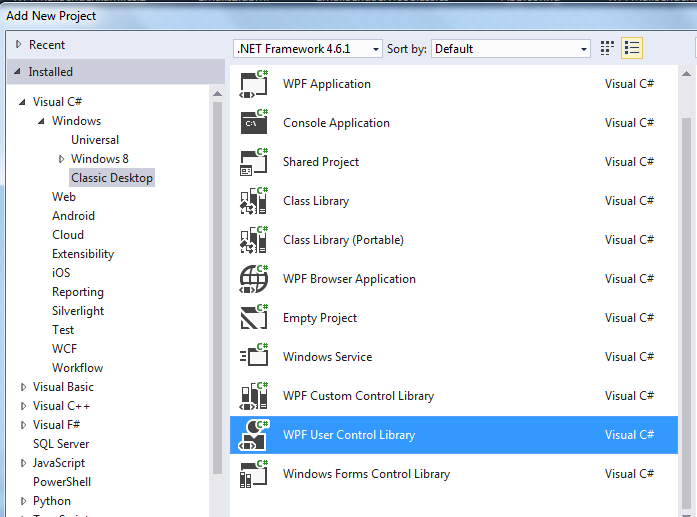


Мы увидим панель «Перейти в планировщик», по клику на которую мы переключаемся на следующую вкладку «Планировщик».

Давайте поменяем эту панель и поместить на это место кнопки позволяющие переключаться между вкладками TabControl’а.

То есть я предлагаю создать свой контрол с двумя кнопками «Следующий» и «Предыдущий», дополнить надписи на кнопках иконками со стрелочками и сделать так, что бы кнопку «Предыдущий» или «Следующий», можно было скрыть при необходимости. По размеру новый контрол будет как панель «Перейти в планировщик». И поместить этот контрол мы сможем на все вкладки TabControl’a.

### Для этой цели выберем проект WPF User Control Library



Иногда не сразу находится этот проект, в моей версии студии он спрятался в Visual C#->Windows->Classic Desktop

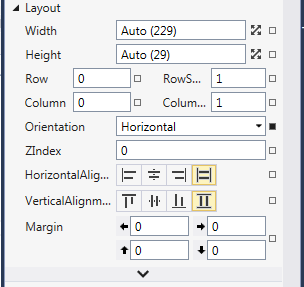
Назовём новый проект TabSwitcher.

Класс, который автоматически создается, назовем TabSwitcherControl.

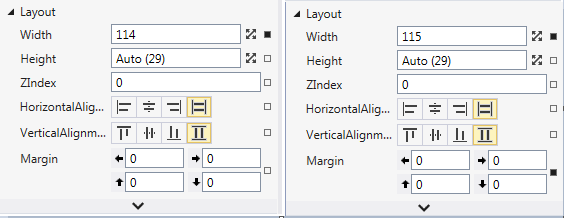
В дизайнере увидим квадрат, куда мы сможем добавлять элементы управления.

Сделаем его по размеру 229 на 30.

Кинем на форму StackPanel, настройки сделаем такие

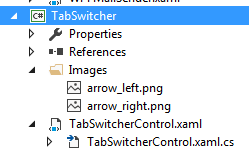


Уже сверху на StackPanel кинем две кнопки



Переименуем их соответственно btnNext и btnPrevious.

Добавим в новый проект папку Images и добавим в неё два файлика с иконками со стрелочками.



Поместим на кнопки надпись «Предыдущий» и «Следующий» и иконки со стрелочками рядом с надписями.

Мы уже добавляли на кнопки StackPanel, что бы поместить на нем и надпись и картинку.

На всякий случай привожу XAML код нашего контрола.

|  |
| --- |
| <UserControl x:Class="TabSwitcher.TabSwitcherControl"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:local="clr-namespace:TabSwitcher"  mc:Ignorable="d"  d:DesignHeight="28.895" d:DesignWidth="300" Height="30" Width="229">  <Grid Height="29" VerticalAlignment="Top">  <StackPanel Orientation="Horizontal">  <Button x:Name="btnPrevios" Width="114" >  <Button.Content>  <StackPanel Orientation="Horizontal">  <Image Source="Images/arrow\_left.png" Width="30" Height="16" />  <TextBlock Margin="3,0,0,0" Text="Предыдущий" VerticalAlignment="Center" />  </StackPanel>  </Button.Content>  </Button>  <Button x:Name="btnNext" Width="115 Margin="0">  <Button.Content>  <StackPanel Orientation="Horizontal">  <TextBlock Margin="3,0,0,0" Text="Следующий" />  <Image Source="Images/arrow\_right.png" Width="30" Height="16" Margin="5,0,0,0" />  </StackPanel>  </Button.Content>  </Button>  </StackPanel>  </Grid>  </UserControl> |

Теперь нужно добавить возможность скрывать кнопки по желанию пользователя. Делаем так, если мы скрываем одну кнопку, то вторая соответственно заполняет весь конрол.

Добавляем вот этот код в наш класс TabSwitcherControl.xaml.cs

|  |
| --- |
| public partial class TabSwitcherControl : UserControl  {  public TabSwitcherControl()  {  InitializeComponent();  }  #region properties  private bool bHideBtnPrevios = false; // поле соответствующее будет ли скрыта кнопка Пердыдущий  private bool bHideBtnNext = false; // поле соответствующее будет ли скрыта кнопка Следующий  /// <summary>  /// Свойство соответствующее будет ли скрыта кнопка Предыдущий.  /// Что бы Свойство отразилось на PropertiesGrid у нашего контрола, его нужно представить именно свойством, а не полем  /// </summary>    public bool IsHideBtnPrevios  {  get { return bHideBtnPrevios; }  set  {  bHideBtnPrevios = value;  SetButtons(); // метод, который отвечает на отрисовку кнопок  }  }// IsHideBtnPrevios  public bool IsHideBtnNext  {  get { return bHideBtnNext; }  set  {  bHideBtnNext = value;  SetButtons(); // метод который отвечает за отрисовку кнопок  }  }// IsHideBtnPrevios  private void BtnNextTrueBtnPreviosFalse()  {  btnNext.Visibility = Visibility.Hidden;  btnPrevios.Visibility = Visibility.Visible;  btnPrevios.Width = 229;  btnNext.Width = 0;  btnPrevios.HorizontalAlignment = HorizontalAlignment.Stretch;  }  private void BtnPreviosTrueBtnNextFalse()  {  btnPrevios.Visibility = Visibility.Hidden;  btnNext.Visibility = Visibility.Visible;  btnNext.Width = 229;  btnPrevios.Width = 0;  btnNext.HorizontalAlignment = HorizontalAlignment.Stretch;  }  private void BtnPreviosFalseBtnNextFalse()  {  btnNext.Visibility = Visibility.Visible;  btnPrevios.Visibility = Visibility.Visible;  btnNext.Width = 115;  btnPrevios.Width = 114;  }  private void BtnPreviosTrueBtnNextTrue()  {  btnPrevios.Visibility = Visibility.Hidden;  btnNext.Visibility = Visibility.Hidden;  }  /// <summary>  /// Метод который отвечает за отрисовку кнопок.  /// Есть четыре варианта, когда обе кнопки доступны. Доступна одна и не доступна вторая. И обе кнопки недоступны  /// </summary>  private void SetButtons()  {  if (bHideBtnPrevios && bHideBtnNext) BtnPreviosTrueBtnNextTrue();  else if (!bHideBtnNext && !bHideBtnPrevios) BtnPreviosFalseBtnNextFalse();  else if (bHideBtnNext && !bHideBtnPrevios) BtnNextTrueBtnPreviosFalse();  else if (!bHideBtnNext && bHideBtnPrevios) BtnPreviosTrueBtnNextFalse();  }  #endregion |

Теперь если мы пересоберем, всё решение и добавим наш новый контрол на нашу панель, то увидим в Properties нашего нового контрола, два свойства IsHideBtnNext и IsHideBtnPrevios, при помощи которых мы можем скрыть одну из кнопок или даже обе.

Теперь нужно добавить возможность, добавлять обработчик по клику на эти кнопки на нашем контроле TabSwitcherControl.

Вначале добавим обработчики на каждую из кнопок в нашем проекте TabSwitcher.

Затем добавим два события (event) для каждой из кнопок.

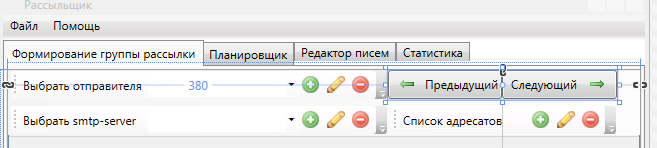


А в обработчиках будем эти события вызывать.

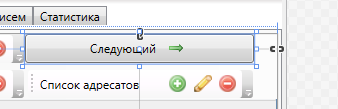
|  |
| --- |
| public event RoutedEventHandler btnNextClick;  public event RoutedEventHandler btnPreviosClick;  private void btnNext\_Click(object sender, RoutedEventArgs e)  {  btnNextClick?.Invoke(sender, e);  }  private void btnPrevios\_Click(object sender, RoutedEventArgs e)  {  btnPreviosClick?.Invoke(sender, e);  } |

Теперь пересоберем все наше решение. И наш элемент управления должен появится в ToolBox’e.

Помещаем контрол на нашей форме. Сразу переименуем его в tscTabSwitcher.

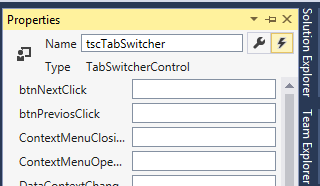


Теперь заходим в свойства контрола в категорию Miscellaneous, находим там свойство IsHideBtnPrevios и ставим для него галочку.



Теперь нужно добавить обработчик по нажатию на кнопку «Следующий».

Если мы зайдем в обработчики событий Properties нашего контрола, то должны увидеть два обработчика.



Но они могут, не проявится в этом месте, до тех пор, пока вы не закроете и не откроете решение. Даже полный Rebuild всего решения может не помочь.

В этом случае можно добавить обработчик события вручную в коде.

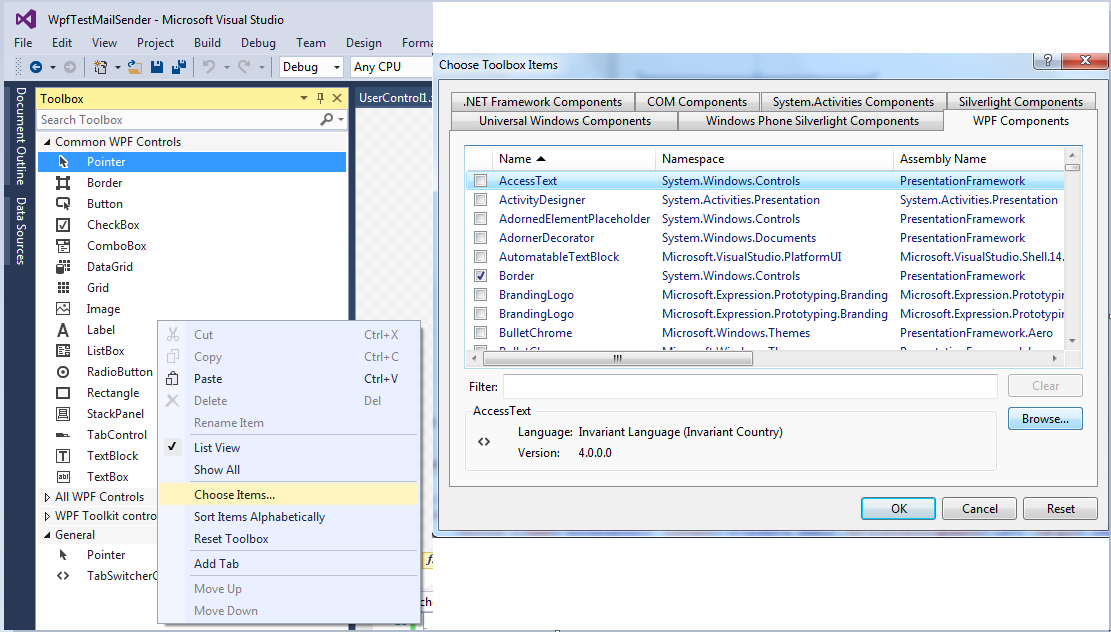
|  |
| --- |
| tscTabSwitcher.btnNextClick += TscTabSwitcher\_btnNextClick;  }  private void TscTabSwitcher\_btnNextClick(object sender, RoutedEventArgs e)  {  tabControl.SelectedIndex = 1;  } |

Теперь в качестве домашнего задания добавьте этот контрол на все остальные вкладки TabControl’а. Выберете нужные кнопки и добавьте обработчики по переключению вкладок.

Теперь давайте проверим, а что будет, если добавить наш контрол в совершенно сторонний проект, не из нашего решения.

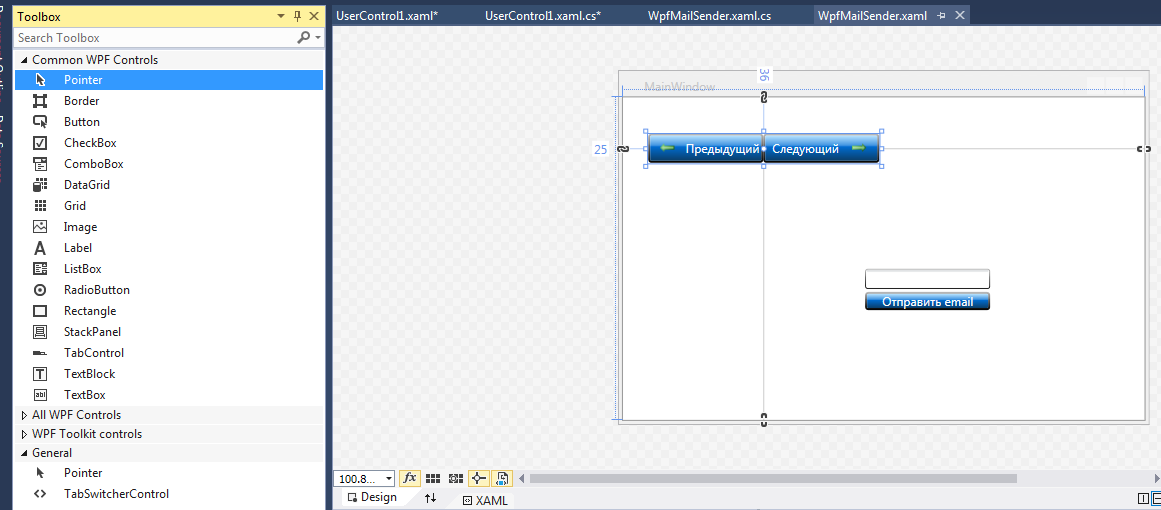
Для этой цели давайте откроем наш тестовый проект, который мы делали на первом занитии, WpfTestMailSender.

Откроем, дизайнер основной формы и кликнем правой кнопкой мыши по ToolBox.



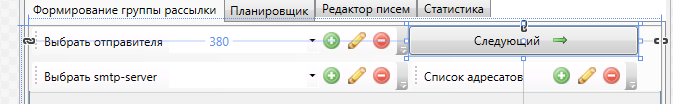
Нажмем Browse и найдем проект соответствующий нашему контролу и выберем DLL.

В ToolBox появится новый контрол TabSwitchControl, который мы можем спокойно кинуть на форму и использовать.



# Домашнее задание.

1. По аналогии с тем как мы создавали DLL из класса, который шифрует пароли, создайте DLL из класса EmailSendServiceClass, который занимается рассылкой писем.
2. Посмотрите на ToolBarTray



Некоторые панели ToolBar очень похожи между собой. Из них можно сделать контрол и добавлять на ToolBar. Сделайте контрол из Панели «Выбрать отправителя» и добавьте его и в качестве контрола «Выбрать smtp-server». У этого контрола должна быть возможность заменить текст у лейбла, должны функционировать комбобокс и все три кнопки.