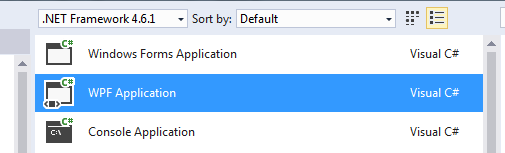
## C# Уровень 3.

# Продвинутый курс с WPF.

# Урок 2

# Введение в WPF. Продолжение.



Введение в WPF. Подключаем основные классы. Прототип приложения.

# Введение.

Сегодня наша цель, это создание общего прототипа приложения. То есть внешнего вида и архитектуры.

Мы добавляем контролы на форму, что бы было понятно, какой внешний вид будет у приложения. Добавляем базу данных с одной таблицей. И добавляем основные классы, при помощи которых будут отправляеться письма.

# Обзор контролов WPF.

WPF имеет в своем составе ряд контролов, которые необходимы для использования в пользовательском интерфейсе. Эти контролы очень похоже на те, что используются в Windows Forms.

## Menu

На прошлом уроке мы добавили меню на нашу форму, теперь добавим пункты меню.

Есть три способа добавить пункты меню.

1. При помощи Properties( в категории Common, пункт Items)
2. Кликнув правой кнопкой мыши по меню и выбрав пунк «Add MenuItem»
3. Прописав в XAML коде

Добавим пункт «Файл» любым из способов. Я предпочитаю правую кнопку мыши, но если вам надо добавить не MenuItem, а какой-либо другой элемент, то лучше использовать Properties.

Что бы добавить к пунктам меню, раскрывающиеся вниз пункты подменю, действуем так же. Любым из вышеперчисленных трех способов добавляем пункты подменю. Добавим пункт «Закрыть». Сразу переименуем его, то есть изменим, свойство Name, назовем miClose. Сразу добавим, обработчик и пропишем внутри него this.Close();.

## TabControl

По умолчанию у контрола с вкладками уже есть две вкладки, которые называются TabItem, давайте добавим ещё две (способы добавления такие же, как и у меню) и переименуем их в:

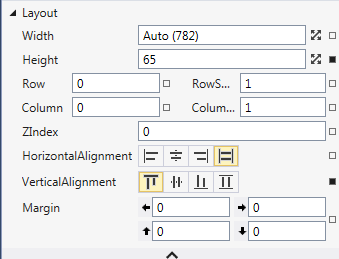
1. Формирование группы рассылки
2. Планировщик
3. Редактор писем
4. Статистика

На первой вкладке «Формирование группы рассылки» у нас будет находиться DataGrid с адресатами и меню с возможностью выбрать отправителя, smtp сервер и редактировать адресаты в DataGrid.

## ToolBarTray

Этот элемент управления упрощает размещение панелей инструментов на одной строке. Контрол ToolBarTray содержит коллекцию элементов ToolBar. Панели инструментов ToolBar можно перетаскивать уже в рабочем приложении, если не устанавливать свойство ToolBarTray.IsLocked = “true” для конкретной панели.

Разместим ToolBarTray во вкладке «Формирование группы рассылки» сверху. И сделаем ширину 65. Длину сделаем AUTO и растянутой по всей длине. VerticalAlignment сделаем Top. Margin сделаем Reset, что бы везде были нули. Таким образом, элемент управления ToolBarTray разместиться на самом верху вкладки с фиксированной шириной и длиной по всей длине окна.



## ToolBar

Элемент управления ToolBar, как правило заполняется другими контролами такими как, кнопки, ComboBox, Сheckbox, RadioButton и др.

Давайте добавим тулбар и снабдим его кнопками и комбобоксом, для выбора почтового ящика отправителя.

Кидаем элемент ToolBar на ToolBarTray. В свойствах устанавливаем фиксированную длину 500 и фиксированную ширину 30. Переименовываем его в tbSender. Кидаем сверху на ToolBar элемент Label, который переименовываем в lSender и в Content прописываем «Выбрать отправителя». Длину и ширину лейбла сделаем фиксированными 130 и 25 соответственно.

## ComboBox

Теперь добавим на ToolBar элемент выбора адреса отправителя. Перетащим мышью данный контрол и кидаем его рядом с лейблом lSender. Сразу переименуем его в cbSenderSelect.

Если вы работали в WinForms, то помните, для чего нужен ComboBox – для выбора конкретного элемента из выпадающего списка. В WPF ComboBox служит для такой же цели. Сделаем фиксированную длину и ширину 270 и 25 соответственно. И добавим два (или более) элемента ComboBoxItem в данный контрол. Для этой цели у нас есть всё теже три способа

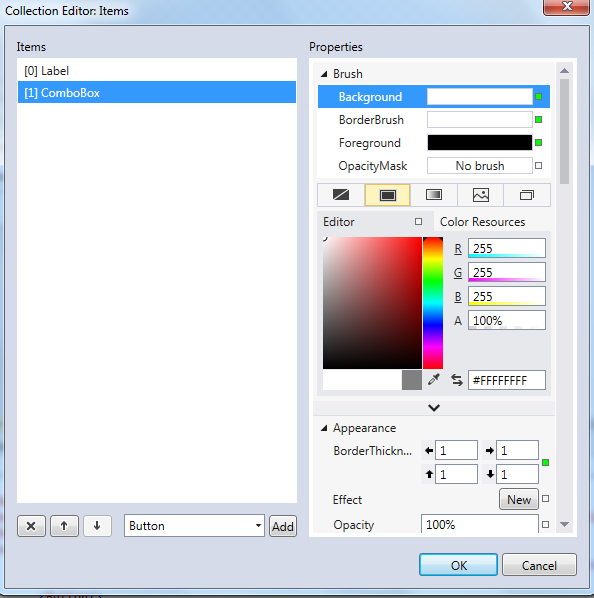
1. При помощи Properties( в категории Common, пункт Items)
2. Кликнув правой кнопкой мыши по меню и выбрав пунк «Add ComboBoxItem»
3. Прописав в XAML коде

В свойство Content каждого Item’a прописываем email отправителя.

В свойстве SelectedIndex элемента ComboBox, мы можем задать тот Item, который будет выбран по умолчанию. И наконец в свойстве ToolTip напишем «Выбрать email отправителя» - эта подсказка будет появляться, если мы наведем курсор мыши на ComboBox.

## Button

У нас есть всё те же три способа добавить кнопку на элемент ToolBar. Давайте для разнообразия сделаем это через Properties. Выберем категорию Common и свойство Items. Появится вот такое вот окно.



Label и ComboBox в списке уже присутствуют. Видите внизу кнопка Add, рядом выпадающий список, через который можно выбрать элемент, какой мы хотим добавить на ToolBar. В принципе, таким же образом мы могли добавить и лейбел и выпадающий список, но их, к сожалению, нет в выпадающем меню, а выбирать их через пункт «другие» очень неудобно. Теперь добавим кнопку, точнее сразу три кнопки.

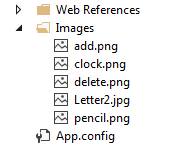
Всем трем кнопкам делаем фиксированную длину и ширину 25 и 25. И делаем их растянутыми.

Первая кнопка – это добавить нового отправителя, соответственно переименовываем её в btnAddSender и в свойство ToolTip пишем «Добавить».

Вторая кнопка – это редактировать нового отправителя, соответственно переименовываем её в btnEditSender и в свойство ToolTip пишем «Редактирвать».

Тертья кнопка – это удалить нового отправителя, соответственно переименовываем её в btnDeleteSender и в свойство ToolTip пишем «Удалить».

Теперь делаем кнопки красивыми. Добавляем в проект папку Images и добавляем в неё картинки, которые нам нужны для проекта.

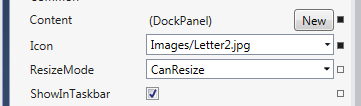


Наша цель поместить красивую иконку на нашу кнопку. Проще всего сделать это через XAML.

|  |
| --- |
| <Button x:Name="btnAddSender" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" Width="25" ToolTip="Добавить">  <Image Source="Images/add.png" HorizontalAlignment="Right" />  </Button>  <Button x:Name="btnEditSender" HorizontalAlignment="Center" VerticalAlignment="Bottom" Width="25" Height="25" ToolTip="Редактировать">  <Image Source="Images/pencil.png"/>  </Button>  <Button x:Name="btnDeleteSender" HorizontalAlignment="Right" VerticalAlignment="Bottom" Width="25" Height="25" ToolTip="Удалить">  <Image Source="Images/delete.png" HorizontalAlignment="Right"/>  </Button> |

Раз мы заговорили о иконках, добавим сразу иконку и на весь проект.

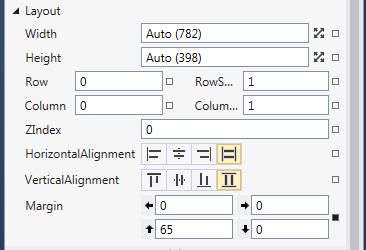
Заходим в свойства главного окна и прописываем в свойстве Icon - Images/Letter2.jpg.



## DataGrid

Этот элемент необходим, для управления визуализацией данных, которые поступают из коллекции объектов (как правило, таблицы базы данных). Выглядит DatagGrid как сетка со строками и ячейками. И он совсем не похож на Grid, который является компоновочным элементом управления.

Кинем его на нашу вкладку и сделаем свойства Layout как на рисунке.



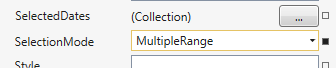
Остальные ToolBar’ы Выбор smtp сервера, переход на вкладку Планировщик и редактирование DataGrid, нужно будет сделать по аналогии и это будет вашим домашним заданием.

## Calendar

Теперь переходим на вкладку «Планировщик» и добавляем на неё элемент календарь. Календарь нам нужен, что бы выбрать даты, на которые будет запланированна рассылка писем.

Поместим календарь на вкладке слева.

Для того, что бы у нас появилась возможность, выбирать сразу несколько дат в свойствах в категории Miscellaneous выберем свойство SelectionMode и установим его в MultipleRange.



По умолчанию календарь выбирает сегодняшнюю дату. Пока мы не будем организовывать сложную рассылку. Ограничемся одним днем. Значение свойства Name для Calendar – cldSchedulDateTimes.

## TextBox

Что бы задать время рассылки, лучше всего подошел бы TimePicker, но его нет в стандартной библиотеке контролов. Что бы его получить надо скачать WPF Toolkit, сейчас заниматься этим не будем. Скачать и установить Toolkit будет домашним заданием. А сейчас разместим TextBox прямо под календарем и сделаем его длину как у календаря. Свойство Name сделаем tbTimePicker.

## Опять Button

Нам нужно две кнопки, которые разместим под TextBox’ом. По длине сделаем их как календарь и текстбокс. Назовем их «Отправить запланированно» и «Отправить сразу» свойства Name зададим btnSend и btnSendAtOnce соответственно.

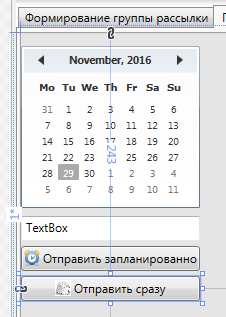
Сделаем такую вещь, добавим маленькую иконку рядом с текстом картинки, благо WPF позволяет это сделать.

Свойство Content позволяет присвоить ему элемент StackPanel, который в свою очередь будет содержать элемент Image и элемент TextBlock. Добавим в XAML соответствующий кнопке «Отправить запланированно» следующий код.

Наша цель поместить красивую иконку на нашу кнопку. Проще всего сделать это через XAML.

|  |
| --- |
| <Button x:Name="btnSend" HorizontalAlignment="Left" Margin="0,213,0,0" VerticalAlignment="Top" Width="179" RenderTransformOrigin="-1.12,-0.727" Click="btnSend\_Click" Height="25">  <Button.Content>  <StackPanel Orientation="Horizontal">  <Image Source="Images/clock.png" Width="16" Height="16" />  <TextBlock Margin="3,0,0,0" Text="Отправить запланированно" VerticalAlignment="Center" />  </StackPanel>  </Button.Content>  </Button> |

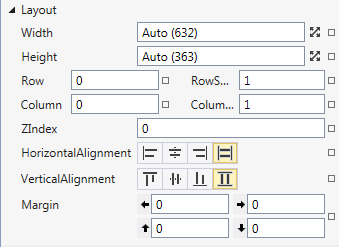
Должно получиться как-то так



Добавить картинку ко второй кнопке, будет домашним заданием.

## RichTextBox

Открываем вкладку «Редактор текста» и просто кидаем на неё элекмент RichTextBox. В свойствах Layout делаем так



На этом визуальная часть нашего прототипа закончена. Можно показывать потенциальному заказчику или начальнику.

# Привязка к данным.

Моё мнение, что прототип, всё-таки должен уметь, что-то делать. Тем более тестовое приложение, которое отправляет письма, у нас уже есть. Осталось только код по отправке писем перенести из тестового проекта в основной проект и адаптировать его. Помимо адаптации кода по отправке писем, надо подключить базу данных с одной таблицей и загрузить email’ы адресатов в проект.

## Привязка к данным ComboBox.

Перед тем, как заниматься данными из базы данных, сделаем более простую операцию, привяжем к нашему ComboBox cbSenderSelect данные по адресатам.

Добавим статический класс, назовем его VariablesClass:

|  |
| --- |
| 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",PasswordClass.getPassword("1234l;i") },  { "sok74@yandex.ru",PasswordClass.getPassword(";liq34tjk") }  };  } |

Здесь мы храним возможных отправителей электронных писем.

Так как я уже говорил, что хранить в одном файле логин ящика и его пароль это очень плохая затея, даже для учебных и тестовых проектов, поэтому добавим в наш проект ещё один статический класс с двумя статическими методами.

|  |
| --- |
| /// <summary>  /// На вход подаем зашифрованный пароль, на выходе получаем пароль для email  /// </summary>  public static class PasswordClass  {  public static string getPassword(string p\_sPassw)  {  string password = "";  foreach (char a in p\_sPassw)  {  char ch = a;  ch--;  password += ch;  }  return password;  }  /// <summary>  /// На вход подаем пароль, на выходе получаем зашифрованный пароль  /// </summary>  /// <param name="p\_sPassword"></param>  /// <returns></returns>  public static string getCodPassword(string p\_sPassword)  {  string sCode = "";  foreach (char a in p\_sPassword)  {  char ch = a;  ch++;  sCode += ch;  }  return sCode;  }  } |

Понимаю, что подобный метод шифровки пароля от честных людей, но как защита от случайного попадания пароля в руки недоброжелателя вполне подходит. Те, кто хочет организовать серьёзную шифровку паролей, можете поискать в интернете, как это делается. Серьезная организация шифрования паролей выходит за рамки нашего проекта, поэтмоу ограничемся вот такой-вот примитивной защитой.

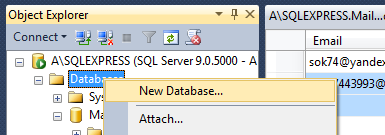
Теперь осталось привязать коллекцию Senders к ComboBox.

Добавляем вот такой вот код в конструктор основного класса нашего проекта WPFMailSender. В файле WPFMailSender.xaml у элемента cbSenderSelect необходимо удалить, ранее добавленные, ComboBoxItem, поскольку указание ItemsSource возможно только для ComboBox без ComboBoxItem.

|  |
| --- |
| public WPFMailSender()  {  InitializeComponent();  cbSenderSelect.ItemsSource = VariablesClass.Senders;  cbSenderSelect.DisplayMemberPath = "Key";  cbSenderSelect.SelectedValuePath = "Value";  } |

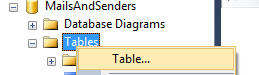
## Привязываем к базе данных DataGrid.

Вначале надо создать базу данных в MS SQL Server. Открываем MS SQL Server Management Studio.



И назовем её MailsAndSenders.

Теперь создаем таблицу

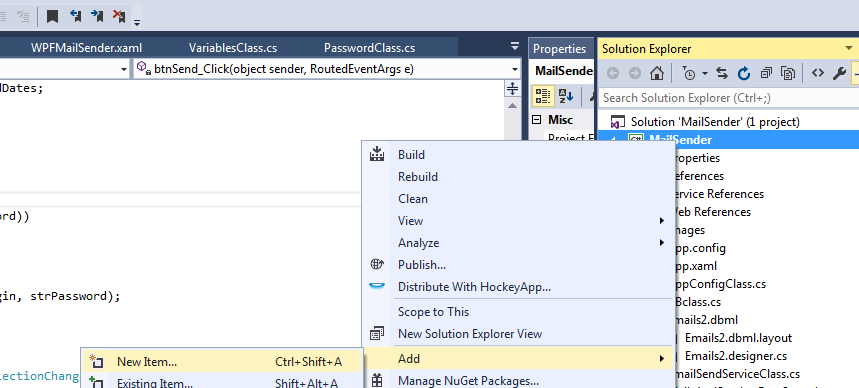


В таблице создаем две колонки: Value ,тип данных nvarchar(MAX) и Name с тем же типом данных. Назовем таблицу Emails.

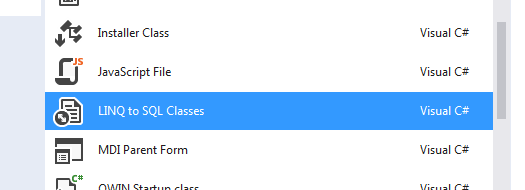
Заполним таблицу тестовыми email’ми и именами.

Теперь надо привязать таблицу базы к нашему проекту.

Заходим в наш проект, кликаем правой кнопкой мыши по проекту и выбираем Add->New Item

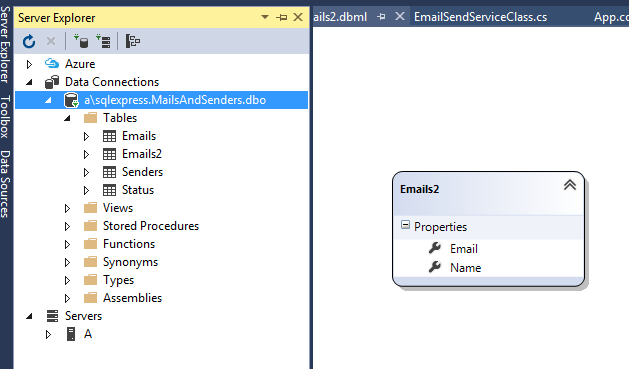


Выбираем Ling to SQL Classes



Назовем новый класс Emails

Открываем полученный dbml файл и просто перетаскиваем на экран мышью таблицу из Server Explorer



Теперь идем в класс который отвечает за базу данных и добавляем вот такой вот код

|  |
| --- |
| /// <summary>  /// Класс который отвечает за работу с базой данных  /// </summary>  public class DBClass  {  private EmailsDataContext emails = new EmailsDataContext();  public IQueryable<Emails> Emails  {  get  {  return from c in emails.Emails select c;  }  }  } |

В конструкторе основного класса теперь добавляем код по привязке данных из базы данных к элементу управления DataGrid, который отвечает за отображение адресатов.

|  |
| --- |
| DBClass db = new DBClass();  dgEmails.ItemsSource = db.Emails; |

# Отправление писем.

Теперь нужно заполнить кодом класс EmailSendServiceClass. Мы почти полностью переносим класс из тестового проекта по отправке писем из прошлого урока. За исключением некоторых изменений.

|  |
| --- |
| class EmailSendServiceClass  {  #region vars  private string strLogin; // email c которого будет рассылаться почта  private string strPassword; // пароль к email с которого будет рассылаться почта  private string strSmtp = "smtp.yandex.ru"; // smtp - server  private int iSmtpPort = 25; // порт для smtp-server  private string strBody; // текст письма для отправки  private string strSubject; // тема письма для отправки  #endregion  public EmailSendServiceClass(string sLogin, string sPassword)  {  strLogin = sLogin;  strPassword = sPassword;  }  private void SendMail(string mail, string name) // Отправка email конкретному адресату  {  using (MailMessage mm = new MailMessage(strLogin, mail))  {  mm.Subject = strSubject;  mm.Body = "Hello world!";  mm.IsBodyHtml = false;  SmtpClient sc = new SmtpClient(strSmtp, iSmtpPort);  sc.EnableSsl = true;  sc.DeliveryMethod = SmtpDeliveryMethod.Network;  sc.UseDefaultCredentials = false;  sc.Credentials = new NetworkCredential(strLogin, strPassword);  try  {  sc.Send(mm);  }  catch (Exception ex)  {  MessageBox.Show("Невозможно отправить письмо " + ex.ToString());  }  }  }//private void SendMail(string mail, string name)  public void SendMails(IQueryable<Emails> emails)  {  foreach (Emails email in emails)  {  SendMail(email.Value, email.Name);  }  }  }//private void SendMail(string mail, string name) |

Появился конструктор, в котором мы задаем логин и пароль отправителя. И метод SendMails в который мы передаем коллекцию объектов класса связанного с базой данных.

## Отправить сразу

Теперь остался последний штрих, прописать сам процесс отправления писем.

Письма будем отправлять по нажатию на кнопку «Отправить сразу» из вкладки «Планировщик».

Создаем, обработчик и вставляем код.

|  |
| --- |
| private void btnSendAtOnce\_Click (object sender, RoutedEventArgs e)  {  string strLogin = cbSenderSelect.Text;  string strPassword = cbSenderSelect.SelectedValue.ToString();  if (string.IsNullOrEmpty(strLogin) || string.IsNullOrEmpty(strPassword))  {  MessageBox.Show("Выберите отправителя");  return;  }  EmailSendServiceClass emailSender = new EmailSendServiceClass(strLogin, strPassword);  emailSender.SendMails((IQueryable<Emails>)dgEmails.ItemsSource);  } |

Теперь если нажать на кнопку «Отправить сразу», письма отправятся по указанным адресам.

## Отправить запланированно

Теперь сделаем последний штрих. Отправка письма запланированно.

Пришло время поработать над классом SchedulerClass, который как раз и отвечает за планировщик.

Добавим две библитеки сразу, одну для таймера, вторую для MessageBox.

|  |
| --- |
| using System.Windows.Threading;  using System.Windows; |

Вот код, который надо добавить в класс SchedulerClass

|  |
| --- |
| /// <summary>  /// Класс планировщик, который создает расписание, следит за его выполнением и напоминает о событиях  /// Так же помогает автоматизировать рассылку писем в соответствии с расписанием  /// </summary>  class SchedulerClass  {  DispatcherTimer timer = new DispatcherTimer(); // таймер  EmailSendServiceClass emailSender; // экземпляр класса отвечающего за отправку писем  DateTime dtSend; // дата и время отправки  IQueryable<Emails> emails; // коллекция email'ов адресатов  /// <summary>  /// Методе который превращаем строку из текстбокса tbTimePicker в TimeSpan  /// </summary>  /// <param name="strSendTime"></param>  /// <returns></returns>  public TimeSpan GetSendTime(string strSendTime)  {  TimeSpan tsSendTime = new TimeSpan();  try  {  tsSendTime = TimeSpan.Parse(strSendTime);  }  catch{}  return tsSendTime;  }  /// <summary>  ////Непостредственно отправка писем  /// </summary>  /// <param name="dtSend"></param>  /// <param name="emailSender"></param>  /// <param name="emails"></param>  public void SendEmails(DateTime dtSend, EmailSendServiceClass emailSender, IQueryable<Emails> emails)  {  this.emailSender = emailSender; // Экземпляр класса отвечающего за отправку писем присваиваем  this.dtSend = dtSend;  this.emails = emails;  timer.Tick += Timer\_Tick;  timer.Interval = new TimeSpan(0, 0, 1);  timer.Start();  }  private void Timer\_Tick(object sender, EventArgs e)  {  if (dtSend.ToShortTimeString() == DateTime.Now.ToShortTimeString())  {  emailSender.SendMails(emails);  timer.Stop();  MessageBox.Show("Письма отправлены.");  }  }  } |

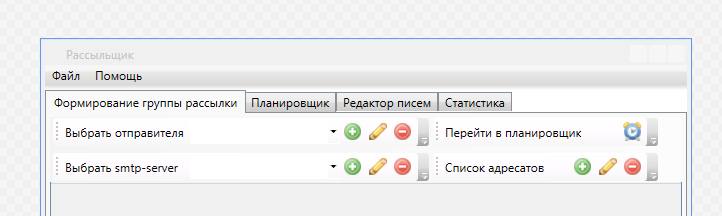
Теперь добавляем код в обработчик кнопки «Отправить запланированно».

|  |
| --- |
| private void btnSend\_Click(object sender, RoutedEventArgs e)  {  SchedulerClass sc = new SchedulerClass();  TimeSpan tsSendTime = sc.GetSendTime(tbTimePicker.Text);  if (tsSendTime == new TimeSpan())  {  MessageBox.Show("Некорректный формат даты");  return;  }  DateTime dtSendDateTime = (cldSchedulDateTimes.SelectedDate ?? DateTime.Today).Add(tsSendTime);  if (dtSendDateTime < DateTime.Now)  {  MessageBox.Show("Дата и время отправки писем не могут быть раньше, чем настоящее время");  return;  }  EmailSendServiceClass emailSender = new EmailSendServiceClass(cbSenderSelect.Text, cbSenderSelect.SelectedValue.ToString());  sc.SendEmails(dtSendDateTime, emailSender, (IQueryable<Emails>)dgEmails.ItemsSource);  } |

# Домашнее задание.

1. В процессе работы, мы создали ToolBarTray и поместили на нём один из элементов ToolBar на который, в свою очередь, поместили лейбл с названием, комбобокс с выбором отправителя и три кнопки. Теперь задание
   1. По аналогии с комбобоксом по выбору отправтилея, надо создать комбобокс с выбором smtp сервера. На лейбле написать «Выбрать smtp-server».
   2. Добавить ToolBar «Перейти в планировщик» без комбобокса с одной кнопкой clock.png
   3. Добавить ToolBar «Список адресатов» без комбобокса с тремя кнопками.

Выглядеть это должно вот так.



По нажатию на кнопку перейти в планировщик, сделайте, чтобы происходил переход на вкладку «Планировщик»

К комбобоксу «Выбрать smtp-server» привяжите Dictionary по аналогии с комбобоксом «Выбрать отправителя». Создайте Dictionary в том же классе, где ключом будет smtp сервер, а значением (с типом int) порт smtp сервера. И сделайте так, что бы значения сервера и порта передавались в экземпляр класса, который отвечает за отправку почты.

1. При отправке писем сделайте проверку, есть ли текст в элементе RichTextBox во вкладке «Редактор писем», если он пуст, то на экран выдавайте окно «Письмо не заполнено» и открывайте вкладку «Редактор писем».
2. Скачайте и установите WPF Toolkit <http://wpftoolkit.codeplex.com/releases/view/617027>
   1. Добавьте вкладку на ToolBox “Wpf Toolkit controls”
   2. Затем кликнете правой кнопкой мыши по “Choose item”
   3. В диалоге, который появился, выберете кнопку “Browse”и выберете dll с Toolkit
3. Во вкладке «Планировщик», замените TextBox для ввода времени, на элемент TimePicker.
4. Добавьте картинку Letter2.jpg к кнопке «Отправить сразу» во вкладке «Планировщик»

## 

# Список использованной литературы

1. Н. Рандольф, Д. Гарднер, М. Минутилло, К. Андерсон. Visual Studio 2010 для профессионалов
2. MSDN