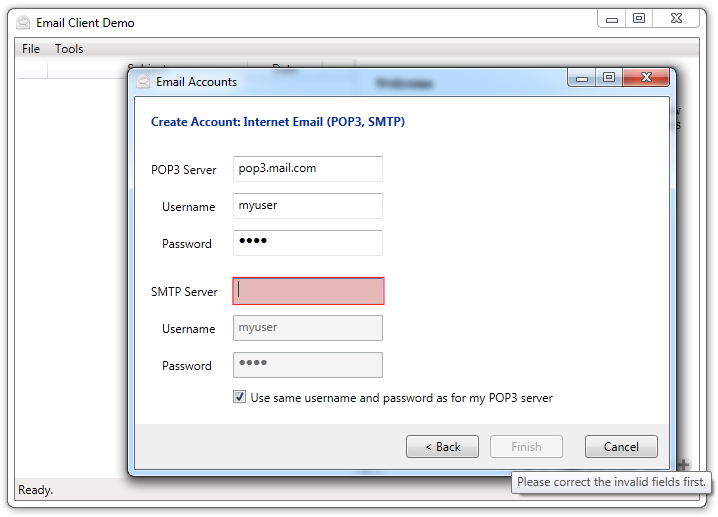
WAF EmailClient

WPF Application Framework (WAF)

# Note: This document is outdated!

# Introduction

The EmailClient sample application shows how to use the WPF Application Framework (WAF) in a wizard oriented application.



This sample application is part of the WPF Application Framework (WAF) [download](http://waf.codeplex.com/).

# Run the sample

1. Open the WpfApplicationFramework solution.
2. Set the EmailClient.Presentation project as StartUp project and start it in Debug mode.
3. Open the Visual Studio Output Window to see the messages from the application.

# Highlights

* Layered Architecture and usage of the Model-View-ViewModel pattern (MVVM).
* UI Workflow (Wizard).
* View composition.
* Showing of a modal dialog.
* Synchronization of data when binding doesn't work (PasswordBox).
* Unit testing of the Application layer.

# Project Structure

EmailClient.Presentation

1. Converters Value converters
2. Resources ResourceDictionaries, Images, Icons
3. Views WPF Views (Windows, UserControls)

EmailClient.Applications

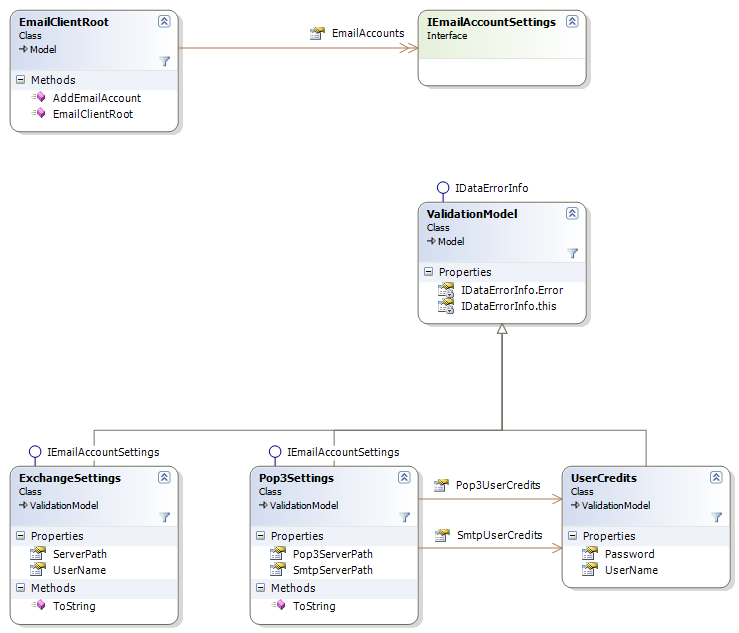
1. Controllers Use case controllers
2. ViewModels ViewModels for the Views
3. Views Interfaces for the Views

EmailClient.Domain

1. (Root) Business objects for the email account settings.

# Domain Layer

EmailClient.Domain\Overview.cd



# Features

|  |  |
| --- | --- |
| UI workflow (Wizard) The ApplicationController implements the EmailAccountsCommand.  EmailClient.Applications/Controllers/ApplicationController.cs  (see RunCreateEmailAccountController method)  Every time the user executes this command a new CreateEmailAccountController is created which is responsible for the wizard workflow.  This controller implements the Back and Next commands.  EmailClient.Applications/Controllers/CreateEmailAccountController.cs  (see Next method)  The Next method checks the current Wizard state and shows the View which comes next.  EmailClient.Applications/Controllers/CreateEmailAccountController.cs (see ShowPop3SettingsView method)  The ShowPop3SettingsView method creates the domain object and the View for the POP3 settings page. |  |
| Advanced UI workflow features The Next button in the wizard changes its text when the last page is shown. Furthermore, a tooltip is shown when the Next button is disabled.  EmailClient.Presentation/Views/EmailAccountsWizard/ EmailAccountsWizard.xaml (see Next button)  The ViewModel contains the information for the Next button which is controlled by the CreateEmailAccountController.  EmailClient.Applications/ViewModels/EmailAccountsViewModel.cs  (see IsLastPage property)  The IsLastPage property is set by the Controller in the Back and Next method. Disabling of the Next button is done in the CanNext method. An update of CanNext is called when the IsValid property changes on the EmailAccountsViewModel.  EmailClient.Applications/Controllers/CreateEmailAccountController.cs (see CanNext and EmailAccountsViewModelPropertyChanged method) |  |
| Validation The wizard validates the input values and shows the boxes in an invalid state when necessary. Additionally, a tooltip shows the validation message.  The Views use Bindings with active validation support.  EmailClient.Presentation/Views/EmailAccountsWizard/ ExchangeSettingsView.xaml (see x:Name="serverPathBox")  The ControlResources file defines the TextBox style with the invalid state appearance and the tooltip.  EmailClient.Presentation/Resources/ControlResources.xaml (see TextBox style)  The ExchangeSettings domain class specifies the validation rules via attributes from the System.ComponentModel.DataAnnotations namespace.  EmailClient.Domain/ExchangeSettings.cs (see ServerPath property)  The ExchangeSettings class inherits from ValidationModel which implements the IDataErrorInfo interface. This interface is used by WPF Binding.  EmailClient.Domain/ValidaitonModel.cs |  |
| Disable command because of validation errors When a validation error occur the Next button of the wizard will be disabled. The CreateEmailAccountController is responsible for the Next command. It listens to the IsValid property of the EmailAccountsViewModel. This shows whether the Wizard is valid or not.  EmailClient.Applications/ViewModels/EmailAccountsViewModel.cs (see IsValid property)  The property is set by the code-behind file of the View.  EmailClient.Presentation/Views/EmailAccountsWizard/ EmailAccountsWizard.xaml.cs (see ErrorChangedHandler method) |  |
| Manual data synchronization Sometimes it is not possible to use Binding for data synchronization. This is the case with the PasswordBox. The application shows how to synchronize the data manually.  The LostFocus event is handled in the SmtpPasswordChanged method.  EmailClient.Presentation/Views/EmailAccountsWizard/ Pop3SettingsView.xaml (see x:Name="smtpPassword")  EmailClient.Presentation/Views/EmailAccountsWizard/ Pop3SettingsView.xaml.cs (see SmtpPasswordChanged method)  The SmtpPasswordChanged method updates the Model class. It is also possible that the property of the Model changes. Then the PasswordBox needs to be updated.  EmailClient.Presentation/Views/EmailAccountsWizard/ Pop3SettingsView.xaml.cs (see SmtpUserCreditsPropertyChanged method) |  |
| Use same username and password The POP3 page let the user to use the same username and password for the SMTP account as for the POP3 account.  The "Use same ..." CheckBox binds on the UseSameUserCredits property.  EmailClient.Applications/ViewModels/Pop3SettingsViewModel.cs (see UseSameUserCredits property)  The UseSameUserCredits property setter already synchronizes the POP3 with the SMTP credits. Furthermore, the ViewModel listens to the PropertyChanged event of the Pop3UserCredits object. When the UserName or Password property changes it updates the SMTP credits.  EmailClient.Applications/ViewModels/Pop3SettingsViewModel.cs (see Pop3UserCreditsPropertyChanged method)  Remark: The ViewModel uses the AddWeakEventListener method to listen for the PropertyChanged events. This way we prevent a memory leak because the Model object lives longer than the ViewModel. |  |