The extensive ApplicationDevelopment Framework that migrates VFX for Visual FoxPro Applications to Silverlight!

Visual Extend 1.0 for Silverlight

Developer Manual

Copyright

Table of Content

Introduction

Visual Extend for Silverlight is an extensive development environment for the development of business applications with Microsoft Silverlight.

In connection with Visual Extend for VFP the VFP developer is presented with the simple task, to migrate existing VFX for VFP projects to VFX for Silverlight. The converted projects have similar behavior in comparison to the original VFX for VFP applications. With VFX for Silverlight the developers discover completely new perspectives.

The migrated Silverlight applications are Internet applications which work with VFP as well as SQL Server databases even if the original VFX for VFP application doesn't use SQL Server database.

Silverlight applications run on Windows and Mac. The number of platforms which support Silverlight is constantly growing. With VFX for Silverlight it is possible to execute code on both the client-side and the server-side.

Development tools

The following development tools are required for VFX for Silverlight:

Visual FoxPro 9 and Visual Extend 13

The migration of existing VFX for VFP projects is realized with the VFX – Silverlight Wizard. To use this assistant, installations of VFP 9 and VFX 13 are required. The VFX – Silverlight Wizard is included in the full version of VFX 13 but not in the trial version.

VFX13 can be downloaded from the website of Visual Extend:

http://www.visualextend.de/

Visual Studio 2010

VFX for Silverlight applications are developed with Visual Studio 2010 and Silverlight 4.

For the development of VFX for Silverlight projects any of the available Visual Studio 2010 versions can be used. It is possible to use the free available version Visual Web Developer Express. Throughout the handbook it is assumed that Visual Web Developer 2010 Express is being used, designated by the general term „Visual Studio“. Other versions of Visual Studio can have more advanced capabilities, which however are not necessary for the development of VFX for Silverlight applications.

Visual Web Developer 2010 Express can be downloaded from the following website of Microsoft:

http://www.microsoft.com/express/Downloads/

Silverlight 4 Tools

After installing Visual Studio, Silverlight 4 Tools also have to be installed.

Silverlight 4 Tools can be downloaded from the following website:

http://www.silverlight.net/getstarted/

Silverlight 4 Toolkit

Silverlight 4 Toolkit can be downloaded from the Codeplex website:

http://silverlight.codeplex.com/

Silverlight 4 Toolkit includes controls, based on the existing Silverlight 4 base classes and offers extended functionalities.

Programming Languages

The development of VFX for Silverlight applications is realized with the C# programming language. Programming with VFP is allowed by using GUINEU. In addition, VFP code can be executed on COM servers.

The user interface is described with the XAML language.

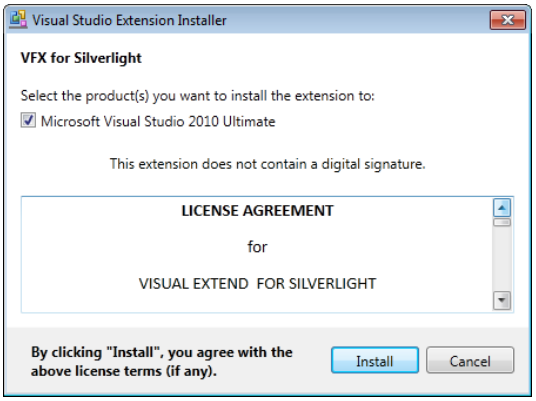
Installation

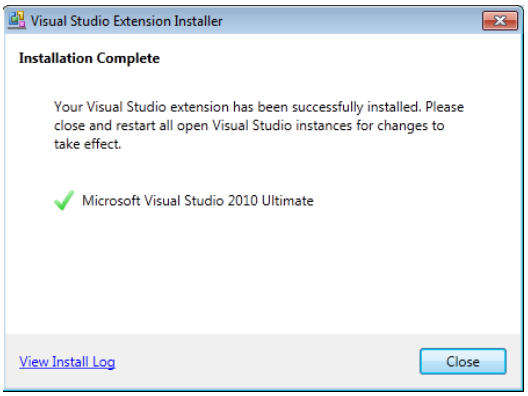
VFX for Silverlight is an extension to Visual Studio and is installed with a Visual Studio Extension Installer.

The extension package has the name VfxForSilverlight.vsix. From VFX for VFP the installation package can be downloaded through the “Update” menu entry and a click on “Get VFX for Silverlight”.

All opened instances of Visual studio have to be closed during the installation.

The installer will appear in the default language of the current Windows installation and detects all installed versions of Visual Studio. You can select the version of Visual Studio that you want to use with VFX for Silverlight.

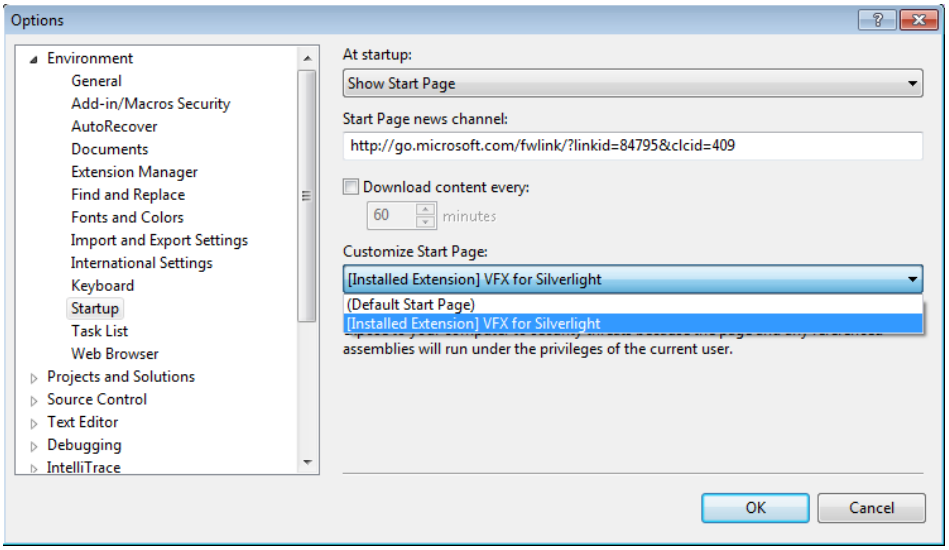




After successfully installing the extension,the template "VFX for Silverlight" appears in the "New project"dialog under the Visual C# category.

Start Page

VFX for Silverlight contains a start page for Visual Studio. It could be set as default start page of Microsoft Visual Studio using Tools/Options dialog.



On the start page of VFX for Silverlight there are options for managing projects identical to these on the start page of Visual Studio. New projects can be created and existing ones can be opened. A list of recently used projects is located on the left.

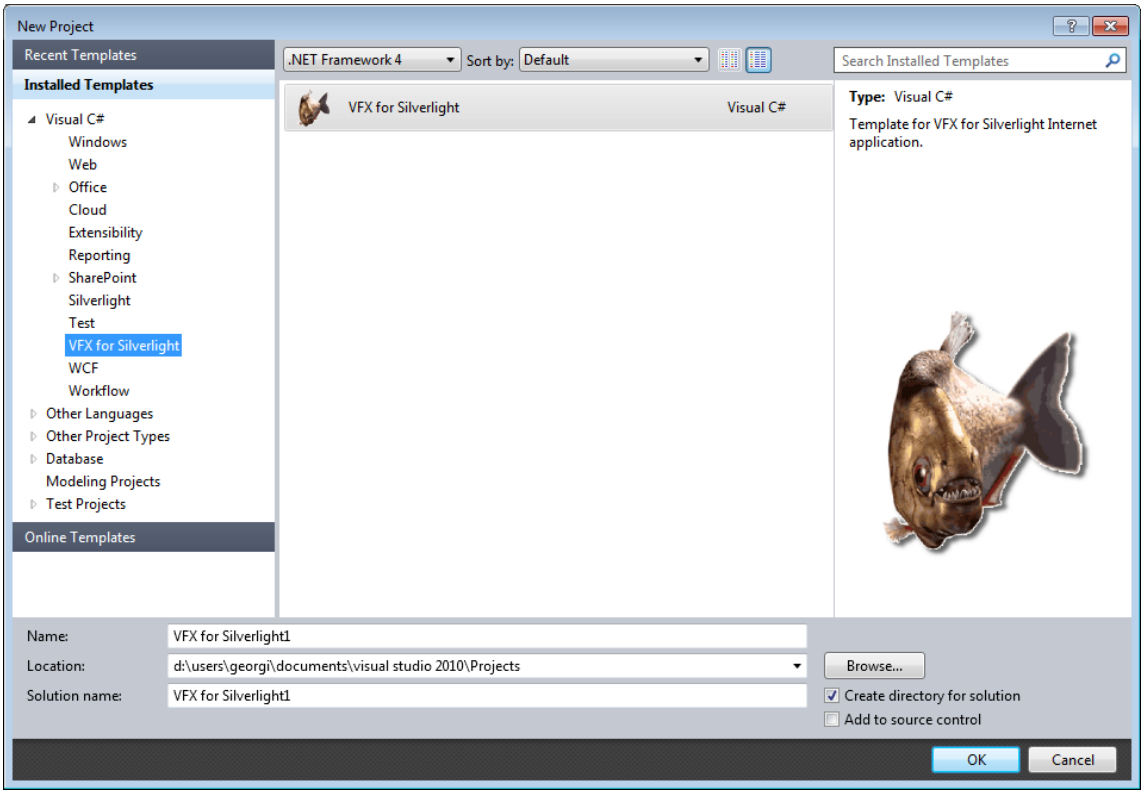
Seven pages with information are on the start page of VFX for Silverlight. They are accessible through the tab buttons. The first page appears at startup and represents the version number of VFX for Silverlight as well as logos with links to the developer, sponsor and distributor of VFX for Silverlight. The information on the following pages appears only if there is an Internet connection.

The second and third page displays the last 50 messages from the German-speaking and English speaking Forum. The following is the portal page of dFPUG. Many files can be downloaded from the portal. The next page shows the current Web page of Visual Extend. Here is a page with links about Visual Extend. The last page shows a ranking of the most active writers in the forum.

Create a New Project

A new project based on VFX for Silverlight can be created from the “New Project…” dialog on the start page or from the “File” menu, “New”, “Project” or by the “New Project” quick button on the toolbar.

On the left side of the “New Project” dialog select “Installed Templates” then the “Visual C#” category. A subcategory “VFX for Silverlight” should appear. Select it. The new project should be given the name of the application which is being migrated. A folder in which to save the new project should be specified. This folder should be provided later in the VFX – Silverlight Wizard.



The project contains a COM server created with VFP. It is ready to be edited with the VFX – Silverlight Wizard. Direct execution is not provided, but yet possible if the VFP COM server is registered.

VFX – Silverlight Wizard

The next development step is the migration of an existing VFX for VFP application to VFX for Silverlight. The project has to be opened with VFP.

If the solution in Visual Studio has to remain open during the VFX - Silverlight Wizard, all files must be saved in Visual Studio before the VFX - Silverlight Wizard starts.

VFX – Silverlight Wizard is integrated in Visual Extend for VFP and can be started via the menu item "VFX 13.0", "Project", "Silverlight Wizard". The assistant allows the migration of forms and reports to VFX for Silverlight. This wizard will guide the developer through the following steps.

The assistant creates and registers a VFP COM server. On Windows Vista and later Windows versions VFP 9 must run explicitly with administrator rights or the wizard will not run properly.

The migrated VFX for Silverlight application accesses the same database as the VFX for VFP application. Each table in the database must have a primary key.

What does the wizard do?

The first step explains the functioning of the wizard. The wizard helps you to migrate an active VFX project into a Silverlight solution.

There has to be a prepared VFX for Silverlight solution to complete this wizard. Through the next steps of the wizard can be selected which forms, features and reports to be migrated.

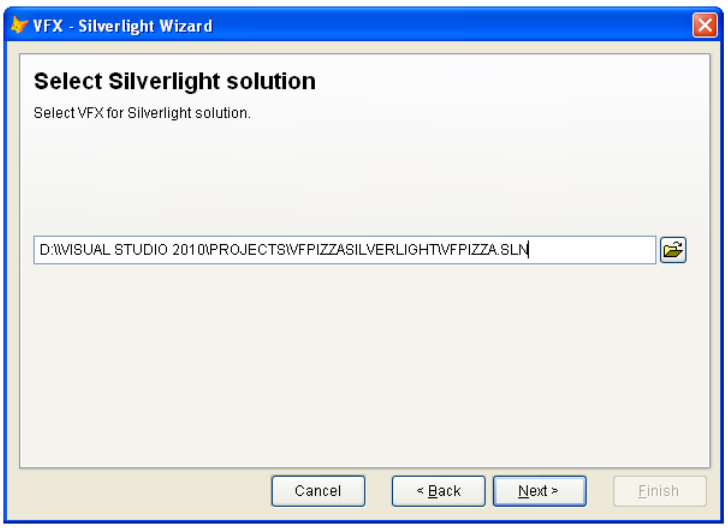
The wizard creates and registers a VFP COM server. VFP 9 must be started with administrator privileges to successfully run this wizard.

The wizard does not affect the VFX for VFP project.

Select Silverlight Solution

In this step a previously prepared VFX for Silverlight solution is selected.

Find the VFX for Silverlight solution and select the .sln file.



The forms and reports which the VFX – Silverlight Wizard generated, have to be added to the selected solution.

Data access

This step adds components for data access to the VFX for Silverlight.

The Silverlight application accesses data through an ADO.NET Entity Data Model and a domain service. Both components are generated by the wizard.

If the wizard is run on the same project again and in the meantime changes to the data structure are introduced, the wizard will generate new ADO.NET Entity Data Model and Domain Service.

The first time the wizard is applied, a VFP COM server is also created. Through the COM Server access to databases is realized. When VFP databases have to be used, the data can be accessed through the COM Server. Business logic created with VFP can also be applied with the help of the COM Server.

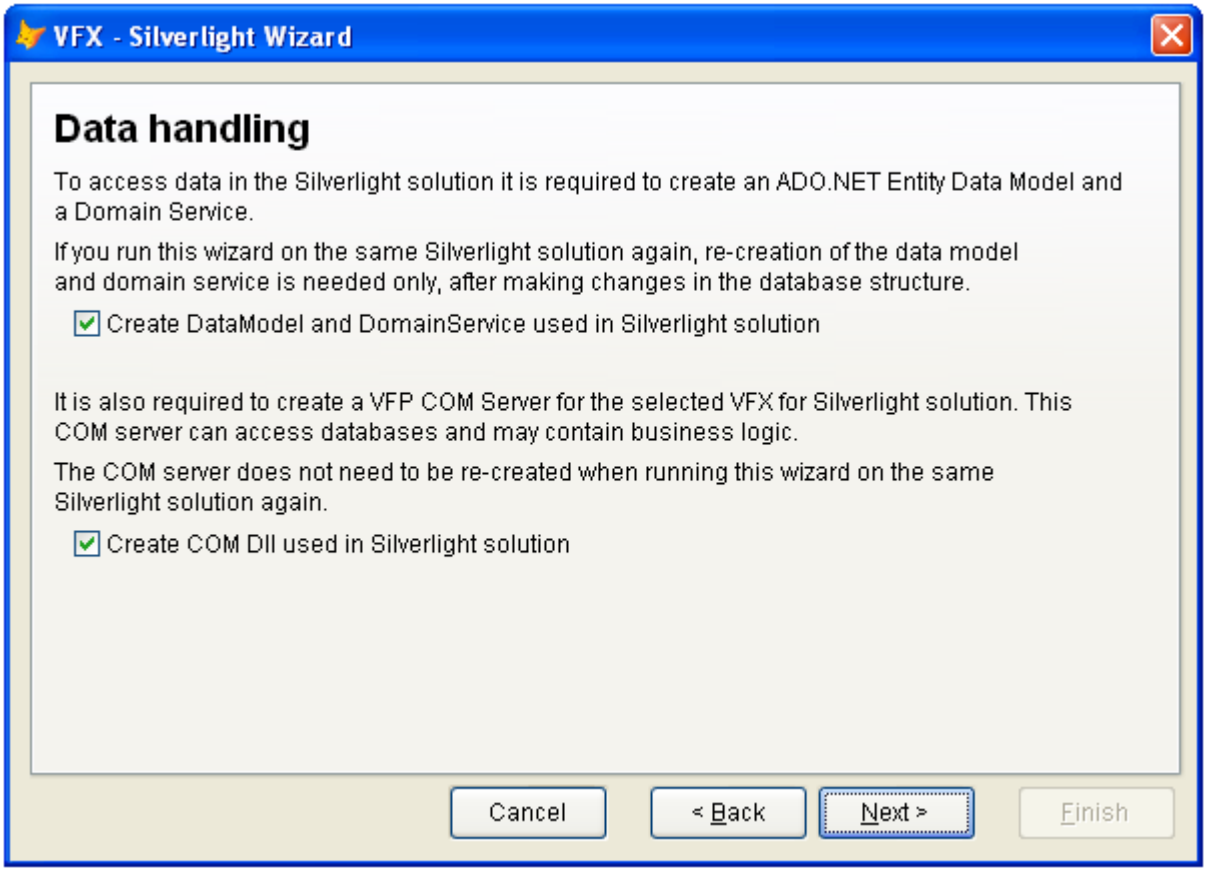
The VFX – Silverlight Wizard looks for the file config.vfx in the VFX project folder. If the file exists, the wizard generates another config.vfx, in which all paths are replaced with full paths. That new file will be copied to VFPComDomainService of the VFX for Silverlight solution. It will be used for data handling via the COM Server Dll. If there is no config.vfx in the VFX project folder, a new config.vfx will be created.

The wizard looks for SQL databases in config.vfx. If one of more are found the client name will be added and connection strings to them will be defined in <application name>.Web\ Web.Config file, too.

Connection to SQL databases will be defined in Web.Config only in case the system tables (VFX tables) are in SQL database, too. In case VFX tables are in DBC the connections to this row will be defined only in config.vfx.

Create DataModel and DomainService used in Silverlight solution - Based on WCF RIA Services technologies database model and service are generated.

The wizard checks the solution for a data model named <database name>Model.cs in the folder VfxDataLayer.Web\Models and data service with the name<database name>DomainService.cs in the folder VfxDataLayer.Web\Services. If one of them does not exist, creation of DataModel and DomainService should be chosen. By default in that case they will be created.



VFX model and service files come with the VFX for Silverlight template project. Application database model and service files are generated by the VFX – Silverlight Wizard.

In the model every database table structure is implemented in a corresponding class with the same name. In service are implemented Get, Insert, Update, Delete and ClearPersistened methods for each table.

For generating model and service is used a SQL database if there is one in config.vfx. If there is not, the database from the first row in config.vfx is used. If there is no config.vfx file, the database included in the project is used.

Nevertheless,the type of database used on generation, generated model and service could handle both SQL and VFP databases if they have the same structure.

It is important the database that is used for model and service generating to have primary keys and relations defined. That primary keys and relations are defined in the generated model and used for database handling.

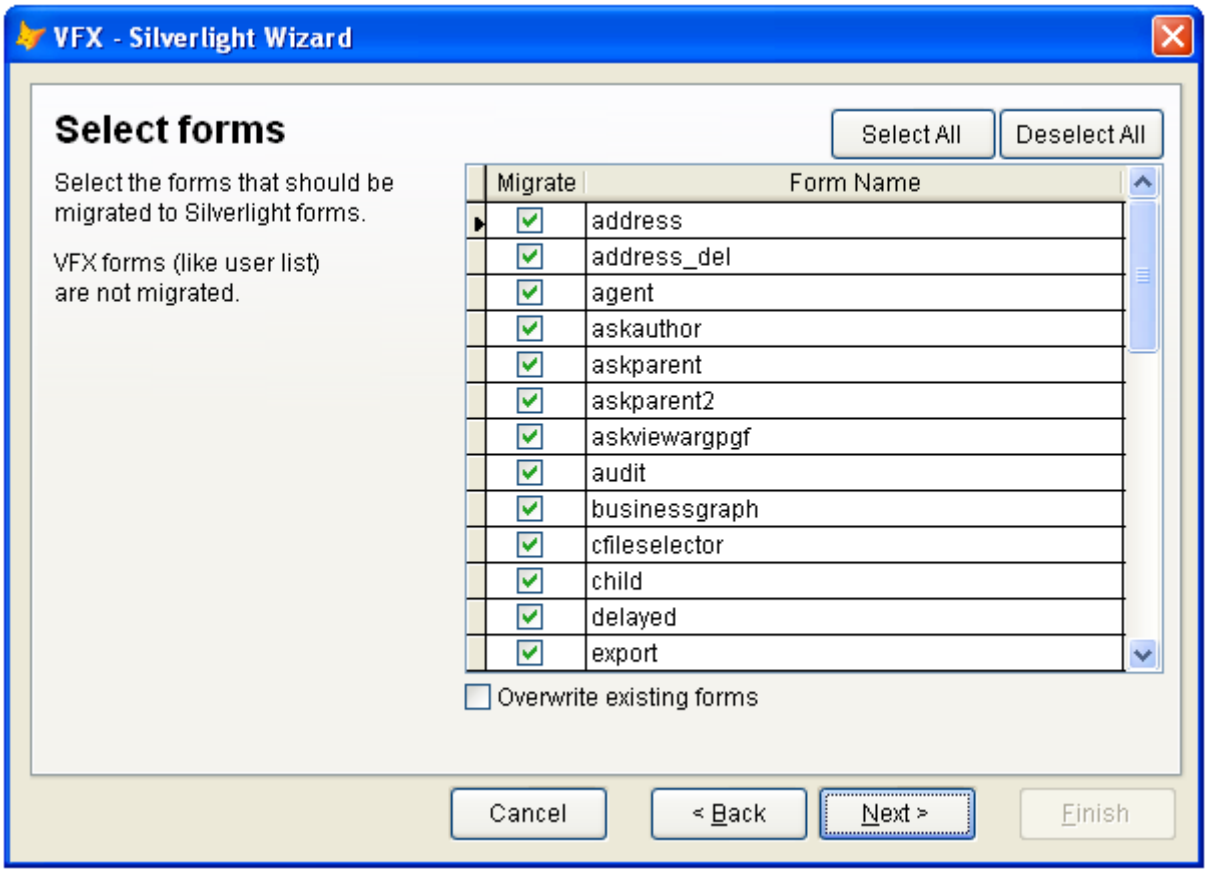
Create COM Dll used in Silverlight solution - If selected, a COM Dll project is created in the folder \VfxDataLayer.Web\VFPComDomainService of the VFX for Silverlight solution. A new Dll will be built and referred as well. The name of the new COM Dll project will be <Project name>COMDomainService. So there will be a specific COM Dll project and specific register Dll for the VFX for Silverlight solution if that option is

chosen. COMDomainService.dll came with the VFX for Silverlight template project and will be unregistered and deleted.

Otherwise COMDomainService.dll will remain to be used.

Form Selection

In this step forms that are included in the VFX for VFP project are listed. By default forms which are included into the VFX for VFP project are selected. VFX forms that are included in the template project are not in the list and are not going to be migrated. An example of such form is the user list.



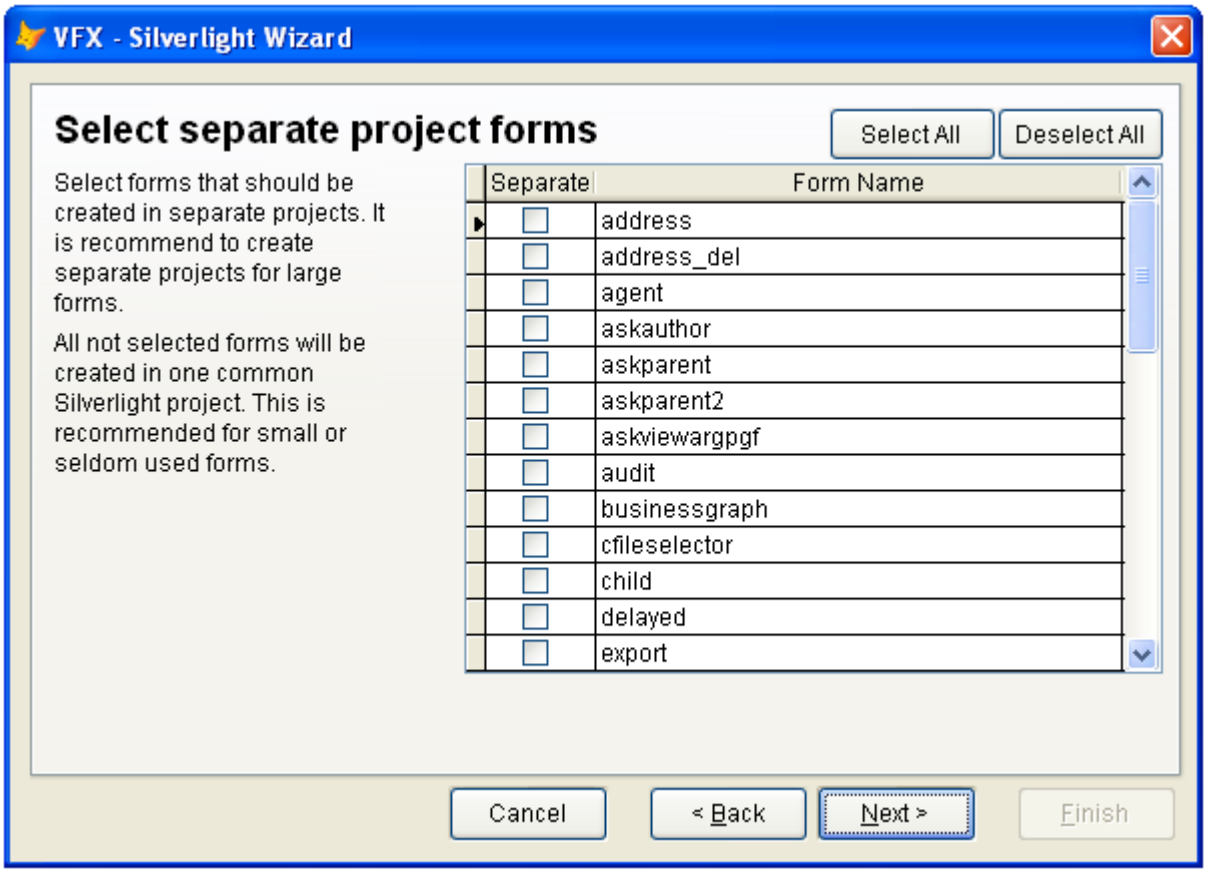
If the data environment of a form uses a data source that does not have a primary key, a warning appears in the log of the wizard.The form can be migrated to Silverlight. The migrated form can run, but the processing of data is not possible as long as no primary key is added to the data source.

Silverlight solution forms with same names will be replaced, if exist.

Selection of projects for the forms

In this step names are going to be given to projects corresponding to the names of the forms which are added. By default all Silverlight forms are included to a project with a name of the type: <Projectname>Forms. It is allowed to choose custom project names. It is also allowed to include any number of forms to one project. When you build the Silverlight client application, a XAP file is generated from each project and that file is transferred to the client through the Internet at runtime. By dividing the application in to separate projects the size of the XAP files is being decreased and consequently the load time is shortened.

Forms can be grouped selectively into projects. It makes sense to put extensive forms in their own projects while smaller forms are grouped together in collective projects. A click on the button „Create separate project for each form“ project generates a name for each form.



New client projects of Silverlight solution will be created for every selected VFP form. Name of new project will be View<form name>. XAML files will be saved in the folder Views of the corresponding Silverlight project.

Other (not selected) Silverlight form files will be stored in the folder Views of <project name> project of the Silverlight solution and will be added to the one project.

That choice is important for the optimization of the new VFX for Silverlight application. Each client project is packed as XAP files. XAP files are used to be downloaded by the client browser. These XAP files are actually .zip files that contain an assembly manifest file and one or more assemblies.

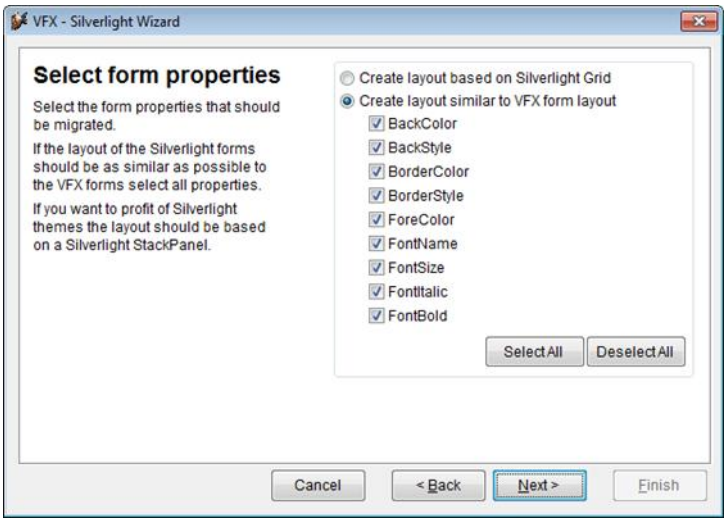
So all unselected forms in this dialog forms will be packed into one XAP file after solution building. If one form has to be loaded from the client browser, the whole XAP file will be downloaded. In case other form from the same XAP file has to be opened, no more information will be downloaded.

All forms in that dialog that are selected will be implemented in different client projects. So each form XAP file will be downloaded when it is needed. If there is only one form in a client project the size of downloaded file will be smaller and no unnecessary information will be downloaded. Other form assemblies will be downloaded only if it needed.

Select Form Properties

By default the Silverlight forms are generated with a layout that closely resembles the layout of the corresponding VFP forms. To achieve this all layout properties are migrated to Silverlight when such migration is possible.

If the user should be able to use styles, not all layout properties should be migrated to Silverlight. All properties which are hard-coded in Silverlight forms cannot be changed by styles at runtime.



"Create layout based on Silverlight StackPanel"–The original position of the controls will not be used.Rather the layout of the generated Silverlight form will be positionedin a Silverlight Grid.

A Silverlight grid is similar to a table. In this table typically column for labels and column for text boxes and other controls for data entry are placed. The advantage of this layout is that forms are scalable. Such forms can be displayed properly on mobile devices with very small screens.

In general for each field in a VFX form there is a label followed by another control like a textbox, or an editbox. These controls belong together. The VFX form builders set the tag property of the label to the same value as the other control has as controlsource. If the VFX – Silverlight Wizard generates the layout based on a Silverlight Grid the connection between the controls is recognized and both controls are put in a row of a Grid. Each control is generated in its own column. So the first column of the Grid will contain the label and the second column will contain the other control like textbox or editbox.

If the user should be able to use styles, not all layout properties should be migrated to Silverlight. All properties which are hard-coded in Silverlight forms cannot be changed by styles at runtime.

The advantage of this layout is that it is better scalable and themes can be applied more easily. There are no layout settings taken from VFP when using this layout.

This feature can be used if you migrate a VFX form created with VFX form builder.

Requirements:

Labels: class – *clabel*

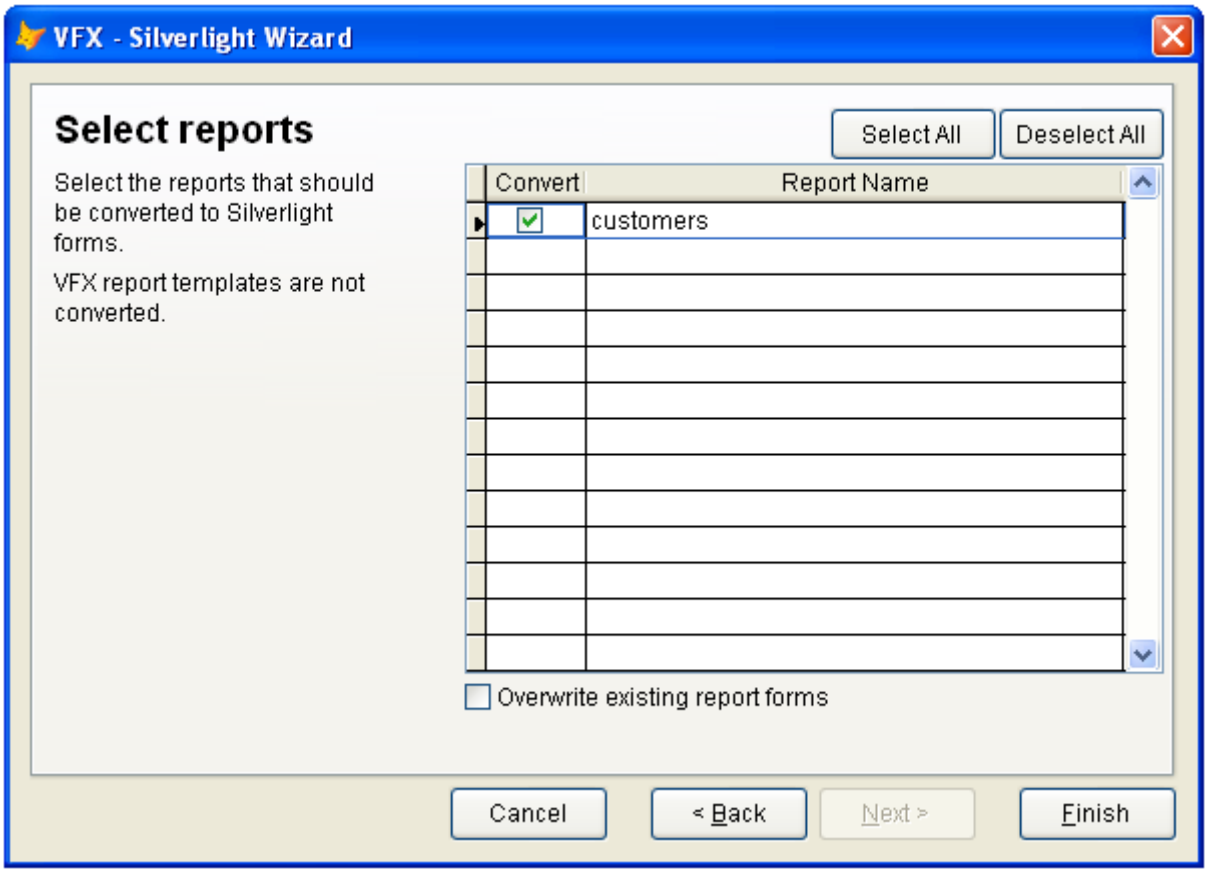
Tag – corresponding with data source property of data control

Data controls: class – *ctextbox, cpickfield, cpickalternate, cpickdate*, others having *ControlSource* property.

Create layout similar to VFX form layout – The layout of the generated Silverlight form will be as similar as possible to the original VFX form. The generated Silverlight form will have a Grid as the outmost container. This container will have just one column and one row. All controls are positioned in this Grid cell using the Silverlight Margin property. This allows absolute positioning of the controls within this cell, similar the use of top and left coordinates in VFP.

Report Selection

In this step in the VFX for VFP shows all project included report files. By default, all reports are migrated except VFX Silverlight reports such as for example the template files for grid reports.



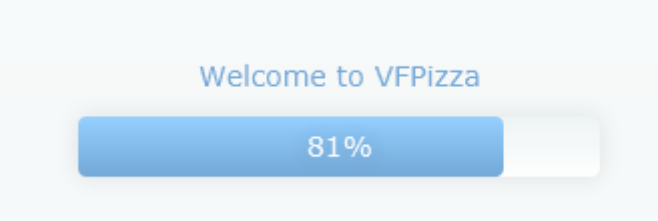
Reports must be based on the data sources provided by the calling form.

If the solution is opened in Visual Studio during the conversion process of VFX – Silverlight Wizard, Visual Studio will detect the ongoing changes in the solution and will open a dialog asking if it should reload the solution. In this case the project has to be reloaded.

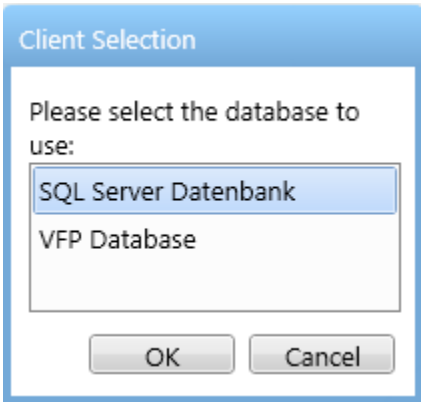
Overwrite existing report forms - Silverlight project report forms with same names will be replaced, if exist.

Start of the New VFX for Silverlight Application

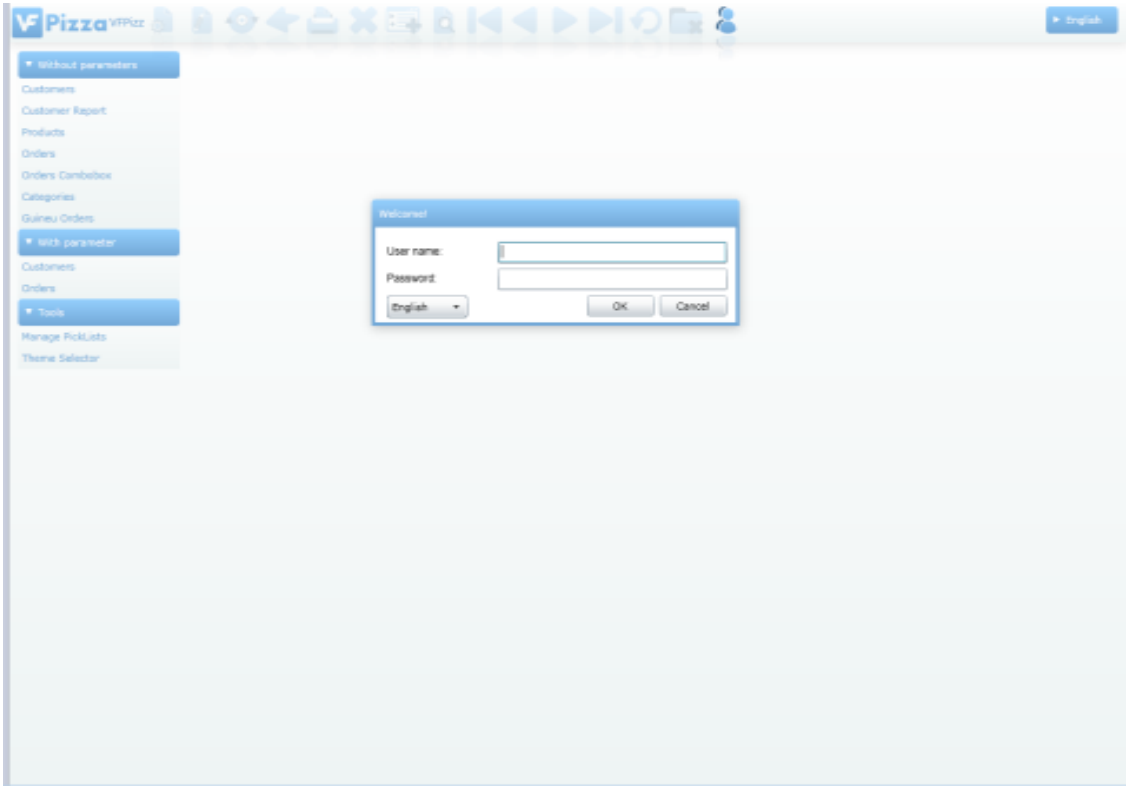
The new created solution can be tested immediately. For this purpose the solution can be started by pressing the green arrow button in the toolbar or by pressing F5.



The application starts in the default Internet browser displaying a splash-screen and a progress bar. If the application is configured to use more than one data source, a client selection dialog appears.



A login window appears. After successful login the application runs. A toolbar appears at the top and left the open dialog appears.



The solution created can be tested immediately. For this purpose the new solution can be started by pressing the green arrow button in the toolbar or by pressing F5. The application starts in the default Internet browser with a splash-screen and with a progress bar. After displaying the splash-screen, the main window is created and the login dialog is invoked. By default each user of a VFX for Silverlight application must be logged in with user name and password. If authentication is not activated, the login dialog is not shown.

Further development with Visual Studio

The VFX for Silverlight application can be processed in Visual Studio according to the specific needs.

It is also possible to modify the forms of the application in VFX for VFP, and then to migrate the modified forms with VFX – Silverlight Wizard.

Architecture

The Silverlight Internet-Application consists of at least two projects: a server project and a client project. The server project is running on the server and contains the data access and business logic.

The client project is the actual Silverlight project and is running on the client computer. The client project is split into many small subprojects for optimization.

Server projects

VFX for Silverlight applications have two server projects.

The startup project of the solution is named VFXforSilverlight.Web. It is running on the Internet Information Server. This project contains the main page default.htm.

The user enters an internet address in his browser to start the application. Default.htm loads and detects if the user has Silverlight plug-in installed and executes the application on the clent-side.

The second server project is named VfxDataLayer.Web. This project contains the data model, that allows access to data, and domain services which allow the exchange of data between client projects and Server projects.

The data access is realized with ADO.NET Entity Data Models. These models can be created with Visual Studio if the database to be used has an ADO.NET Entity Data Model driver. Such driver should be already available for Microsoft SQL Server.

During the migration of VFX for VFP projects an ADO.NET Entity Data Model for the access to VFP Databases is created. The access to the database in this case is accomplished with a VFP COM server that is also generated from the VFX - Silverlight Wizard.

A domain service can be generated using the Visual Studio wizard. By migration of a VFX for VFP application to Silverlight a domain service is created by the VFX - Silverlight Wizard.

Client projects

When compiling a Silverlight project a file with the name extension .XAP is created. In fact the file is a ZIP archive of the DLLs that contain the logic and resources of the application. By renaming the extension of the file to ZIP the contents can be examined.

The user types the address of the application in his browser. The main page contains the name of the initial XAP and the Plug-in version required. Then the browser of the user downloads the initial XAP file over the Internet and executes it with its Silverlight Plug-in. If the required Plug-in version is not present the user is directed to the Silverlight download page.

VFX for Silverlight is built by the module concept. The application is executed with loading the file VfxLoader.xap. Subsequently VfxMainScreen.xap and VfxForm.xap are loaded.

Class hierarchy

The class hierarchy implemented in VFX for Silverlight is similar to the hierarchy that was implemented in VFX for VFP.The functionality of VFX for Silverlight is contained in class files with the prefix "Base". From these class files one to one inheritance exist in class files without the prefix “Base”. Custom extensions can be integrated in the latter mentioned class files.

Application Object

The application object is defined in the VfxApplicationBase project in the program file Controls\VfxAppObjectBase.cs. A one-to-one derivation for individual adjustment exists in the VfxApplication project in the class file Controls/VfxAppObject.cs

The application object provides global properties and methods. Settings can be applied to the properties of the application object that will affect the behaviour of the entire application.

Application settings

AllowRelogon- Allows of current user to login again without leave the application.

AllowUserCustomization – If the value of this property isset to true, logged-in users can add individual settings to the application. If no user is logged, no individual settings can be made. When the value of this property is set to false, no user can add individual settings.

ApplicationName - This property specifies the name of the application. The name appears in the toolbar next to the icon of the application.

AskToSave - Determines if there will be MessageBox asking for saving prior to some operations.

AutoEdit- Determines if a form will enter edit mode on interactive change of controls. Has the following setting: AllwaysEnabled, AllwaysDisabled, UseObjectSettings

AllwaysEnabled-Controls are enabled by default on all forms. The user can set the focus to a control and immediately begin editing. With the first user inflicted change the form automatically switchesfrom view mode to edit mode.

AlwaysDisabled

This setting makes the controlsdisabled by default on all forms. Editing is possible only after the user clicks on the "Edit" icon on the toolbar. Then the form switches to edit mode view.

UseObjectSettings

The settings of the respective form are used.

Century - This property set the format of the century in date fields. If this property is false, the century is missing, the year appears in a double-digit rate form (yy). If this property is true, the year is in four digit form (yyyy).

DisableFormResize – When the value of this property is set to true, users cannot change the size of the forms at run time. If false, the forms are resizable at run time. During the closing of the form the user selected size is saved in VfxResources and restored the next time the form is opened.

EditDateFieldName - The date of the last edit of a record.

The name of the field which stores the date of last modification of a record. If a field with the specified name is in a table, the field is updated at each save. The type of the field can be date or DateTime. If the type is DateTime a timestamp is inserted when saving.

EditTimeFieldName – The name of the field that stores the time of last modification of a record. If a field with the specified name is in a table, the value is automatically populated every time you save. The type of the field must be C(8).

EditUserFieldName - The name of the field in a table that stores the name of the user who last edited the record. If a field with the specified name is in a table, the value is automatically populated each time you save. The field must be of type char length 32.

EnforcePasswordHistoryCount = 0;

This property determines how many old passwords are stored in history. Passwords that are in history can’t be assigned as a new password. The default value is 0.

ErrorDetailLevel –Determines how detailed the log of an encountered exception will be. The exception log is entered in the VfxLog table. It has three levels:MessageOnly,CallStackInformation,FullDetailedInformation

HideWhenEmpty - This property can determine whether controls are hidden when no records are stored in a table. In this case, a note appears on the form. When the userclicks on this note, the form is switched to new entry mode.

InsertDateFieldName - Contains the name of the field which stores the date of the creation of a record. If a field with the specified name is in a table, the field is filled automatically during the first save of a record. The type of the field can be date or DateTime. If the field is of type DateTime a timestamp is inserted when saving.

InsertUserFieldName-The name of a field in a table that stores the name of the user that has created the record. If a field with the specified name is in a table, the value is filled automatically during saving. The field must be of type char with length 32.

InsertTimeFieldName - The name of a field that stores the time of the creation of a record. If a field with the specified name is in a table, the value is filled automatically during the first saving. The type of the field must be C(8).

LangID – If runtimelocalization is not used, the abbreviation of the default language is specified in this property. The default is English.

LoginBehavior - This property specifies whether a user login is required at startup, optional or not required. It determines the startup behavior of the application.

Optional login (default behavior)

The application starts without login. It uses the default database. This is the first database that is found in the Web.config file. When the database is registered in the Web.config file, the first database in the file Config.vfx is used. While the application is running, a login by clicking on the "Login" in the toolbar button is possible. This setting is suitable for applications where some functionality will be accessible to the public, while other parts of the application are protected by user login. This way Internet-applications are made possible, which are read-only accessible without login while a write access requires a login.

LoginAtStartup

The application starts with user login. This behavior is the same of VFX for VFP applications. If the Web.config and Config.vfx files found more than one database, the client selection dialog appears before the user logon. The application doesn't start without user login. This is recommended for Internet and intranet-applications, where no functionality publicly or without registration, should be available.

NoLogin

The application starts without login. It uses the default database. This is the first database that is found in the Web.config file. If no database is registered in the Web.config file, the first database in the file Config.vfx is used. A user login is not possible. This setting is recommended when an Internet application should be available to the open public or not-public intranet applications where user login should not be used.

MainForm - The form specified in this property opens automatically when the application starts. If user login is required, the form will open after login.

MultiInstance - Determines globally if forms can be open more than once.

PasswordHistoryCount - Determines the number of unique new passwords that have to be associated with a user account before an old password can be reused.

Value must be between 0 and 99.

PasswordLength - This property specifies the minimum length of passwords. If this property is set to 0, a password is not required.

PasswordStrengthLevel – Indicates the strength level of required password. Has four levels of security.

Weak- no requirements;

Medium- The password must contain characters from minimum 2 of the following 4 categories:

English uppercase characters (A through Z),

English lowercase characters (a through z),

Base 10 digits (0 through 9),

Non-alphabetic characters (for example,!, $, #, %).

Strong – The password must contain characters from minimum 3 of the above mentioned 4 categories.

Best – The password must contain characters from all of the above mentioned 4 categories.

PasswordValidityDays - This property sets the period of validity of passwords. If a password has expired, prompts the user during login for registering a new password. Login is not possible without a new password. If this property is 0, passwords remain valid indefinitely.

PathToApplicationLogo - Path to application logo.

Defaultpath: "/VfxLoader;component/Icons/Icon128.png"

RelogonQuit –Determines if the browser should be closed if the user clicks cancel on the login dialog.

RequiredFieldInitStyle - This property sets the style of the required fields. The default is "Required". Other custom styles can be implemented.

RequiredFieldFailureStyle -This property sets the style of the mandatory fields if they are empty when saving is initiated. The default value is "RequiredInvalid Other custom styles can be implemented.

RuntimeLocalization When the value of this property is set to true, the application localization executes at runtime. All text to be displayed is read at run time from the Vfxmsg table and displayed in the selected language. When the value of this property is set to false, no texts from table Vfxmsg are read at run time

SaveFormLayoutResolutionDependent - Defines if a form's layout should be saved depending on the user resolution.

ShowFilterActivatedInFormCaption - Indicates if message is shown in Form Title when Filter is applied.

ShowNTLogonFieldInUserManagement – When true Ntlogonfield is displayed in User List.

ShowIntroForm - This property determines whether to display the splash screen.

Methods

GetParametersForLoadLocalizationData(string language, Guid guid)

Create an array with specific data and structure. The array is used to load an active language for application or load specific key from table vfxmsg, which are used to localization.

<param name="language">Language.</param>

<param name="guid">GUID.</param>

returns>Array of strings with specific structure and data.

In element with index 0 is store 'language' parameters.

In element with index 1 is store 'guid' as string.

In element with index 2 is store a value that indicate whether

use COM or SQL Databases.

In element with index 3 is store a value that indicate client name.

```
public virtual void VfxMessageBox(string message, VfxMessageBoxButtons
buttons,
                                                  VfxMessageBoxTypes
messageBoxType,
                                                  string caption, int timeout,
                                                  Action<VfxMessageBoxResults>
response
                                                  )
```

Shows a Message Box

<param name="message">Message to show

<param name="buttons">Type of buttons in the MesssageBox

<param name="messageBoxType">Type of the MessageBox

<param name="caption">Caption

<param name="timeout">Timeout to close MessageBox with MsgResult.NoResult, 0 for no timeout

<param name="response">Callback which is called after closing the MessageBox

```
public static bool RunForm(string formName, string xapName, object[]
args,
            Action<object, EventArgs> callBack)
```

Opens a form from a specified XAP file and name.

```
<param name="formName">The form name.</param>
```

```
<param name="xapName">XAP file name.</param>
<param name="callBack">Callback function.</param>
<param name="args">
```

An array of arguments that match in number, order, and type the parameters

of the constructor to invoke. If args is an empty array or null, the constructor

that takes no parameters (the default constructor) is invoked.

<returns>Return true when user has permit to open form, in other case return false.

```
<exception cref="System.NullReferenceException">
```

When can't find object of IVfxFormActivator in Application.Current.Resources.

```
public static void ExitApplication()
```

Exit from application.

```
public static void SetBrowserProperties()
```

Set browser properties.(title)

Execute Service Operations:

```
public static InvokeOperation ExecuteServerInvokeOperation(
            DomainContext domainContext,
            string invokeOperationName,
            Type returnType,
            IDictionary<string, object> parameters = null,
            bool hasSideEffects = true,
            Action<InvokeOperation> callback = null,
            object userState = null)
```

Forms

```
Forms have properties that control the behavior of the form.
```

Properties

Properties of forms are specified in the file with the extension.xaml.cs of a form.

AskToSave = true;

When the value of this property is set to true, the user is asked to save changes when attempting to close the form and any unsaved changes exist.

AutoEdit = true;

If the value of this property is set to true, the controls on a form are enabled. The manipulation of data can be started immediately.

If the value of this property is set to false, the controls on a form are disabled. If the user wants to edit the data, he must first click the "Edit" button to toggle the form in edit mode and the controls will switch their mode to “enabled”.

AutoResizeControl = true;

Specifies if the controls are resized automatically by the parent form.

Execution of VFP Code

GUINEU

VFP code can be executed in all Silverlight client projects with help of GUINEU. GUINEU is a runtime environment which runs executable files compiled with VFP.

The GUINEU runtime environment is hosted in the file vfx.guineu.runtime.dll. This DLL is located in the project VfxGuineuRuntime. In the same project is placed also the class VfxGuineu.cs which enables the functionality of GUINEU.

The code has to be created with the development environment of VFP. The compiled FXP file must be included in the respective Silverlight project.

It is recommended that the PRG file is also included to the Silverlight project. By double-click on the PRG file in Solution Explorer VFP will start and the file can be edited.

In order to use GUINEU in a C# class a field of type VfxGuineu must be added. The instance is given the name of the compiled VFP file as a parameter.

```
private readonly VfxGuineu _fox = new VfxGuineu("Orders.FXP");
```

The calling of functions in the VFP executable file is possible with a single line of C# code.

```
<Output> = _fox.Do("<function name>", <optional parameter>, ...);
```

Es können so viele Parameter übergeben werden, wie als Parameter von der VFP Funktion akzeptiert werden. Parameter können von beliebigem Typ sein. Es ist auch möglich Objekte aus der Silverlight Benutzeroberfläche als Parameter zu übergeben. Die Eigenschaften von Objekten stehen in der VFP Funktion zur Verfügung und können gelesen und geschrieben werden.

The evaluation of the return value is optional. As many parameters can be passed as the VFP function can accept. The parameters can be of any type. It is also possible to pass objects from the Silverlight user interface as parameters. The properties of objects are available in the VFP function and can be randomly accessed.

```
private void xpgfPageFramePage1txtShiptoname_GotFocus(object sender,
System.Windows.RoutedEventArgs e)

{

    _fox.Do("ShipToName_GotFocus", txtTooltip);

}


private void xpgfPageFramePage1txtShiptoname_TextChanged(object sender,
System.Windows.Controls.TextChangedEventArgs e)

{

    OnEdit();

    _fox.Do("Validate", xpgfPageFramePage1txtShiptoname,
xpgfPageFramePage1edtShiptoaddress, chkSpeichern);

}


private void xpgfPageFramePage1txtOrderdate_LostFocus(object sender,
System.Windows.RoutedEventArgs e)

{

    OnEdit();

    var orderDate =
((orders)ViewModel.WorkAliasCollection.CurrentItem).orderdate;

    txtDelivery.Text = _fox.Do("Lieferdatum", orderDate);

}
```

Passing parameters

There is a method ExecuteFunction in the COM server. Via this method can be executed all native one parameter VFP function. The result is returned as a string. This COM method can be used in two ways in the VFX for Silverlight solution – directly using of VFXDomainContext service method ExecuteCommandService or using special developed VfxControl:VfxActionButton control.

Direct using of the VFXDomainContext Service Method ExecuteCommandService

In the following example the COM server executesthe VFP proper() function and returns the result. This result is get via a callback Callback() function and is set to the variable "propername".

To execute the COM server procedure is used the VFXDomainContext service method ExecuteCommandService. It is called via InvokeOperation.The callback function gets the result.

So we pass the procedure name "proper" and procedureParmater "jack jackson" and get the result "Jack Jackson" in the variable propername.

```
using VfxDataLayer.Web.Services;
using VfxApplicationBase;


namespace Test
{
   publicpartialclassTest : VfxObject
   {
      string propername = "";


privatevoid btnProper_Click(object sender,
System.Windows.RoutedEventArgs e)
      {
         VFXDomainContext _VFXDomainContext = newVFXDomainContext();


         IDictionary<string,object> parameters = newDictionary<string,
object>();
         parameters["procedureName"] = "proper";
         parameters["procedureParmaters"] = "jack jackson";
         parameters["clientName"] =
VfxAppObjectBase.CurrentConnectionInfo.ClientName;
         _VFXDomainContext.InvokeOperation("ExecuteCommandService",
typeof(string), parameters, false, Callback, null);


      }
      publicvoid
Callback(System.ServiceModel.DomainServices.Client.InvokeOperation op)


      {
         propername = op.Value.ToString();
      }
}
}
```

Using special developed VfxControl:VfxActionButton control

Using the VfxActionButton class can be done similar to the first example. There is used bound VfxActionButton property that is used internally as function property on VfxActionButton.click.

First, VfxControl:VfxActionButton control is dropped from the Toolbox container to VFX for Silverlight from.

```
<VfxControl:VfxActionButton
   Content="Action"
Visibility="Visible"
   Height="23"
   HorizontalAlignment="Left"
   Margin="370,209,0,0"
   Name="btnAction"
   VerticalAlignment="Top"
   ToolTipService.ToolTip="Use VFP function PROPER() to format Customer
Name"
                      Width="75"/>
<VfxControl:VfxComboPickList
```

Then the `btnAction` properties are set in the form constructor and `Callback function is defined`:

```
publicpartialclassCustomers : VfxForm.VfxDataForm
{
   public Customers()
   {
.....
btnAction.ProcedureName = "proper";
btnAction.SetBinding(VfxActionButton.ProcedureParmatersProperty,
newBinding { Source = customersViewModel, Path =
newPropertyPath("Customers.CurrentItem.customername"), Mode =
BindingMode.OneWay });
            btnAction.Callback = Callback;
.....
}


.....
publicvoid
Callback(System.ServiceModel.DomainServices.Client.InvokeOperation op)
      {
   ((customers)((CustomersViewModel)ViewModel).
Customers.CurrentItem).customername = op.Value.ToString();
         this.FormStatus = VfxApplicationBase.FormStatus.EditMode;
      }
.....
}
```

Instead of using the function proper() there can be used any function developed in VFP.

VFP developed functions execution

VFX for Silverlight Wizard can change vfpcomdomainservice.dll with application Com Dll named <Project name>COMDomainService.dll. In that case there is created a COM server project in \VfxDataLayer.Web\VFPComDomainService folder.

There are two ways to use VFP code.

1. Use ExecuteFunction method of the main vfpCOMDomainService class.

The function accepts exactly one parameter and returnsa string. If the function execution is not successful, the function returns an error message in the following format: “Error: ” + message()

A new developed function can be executed via the method ExecuteFunction. In this case it is required to have one parameter and a return string as result. If the result type is different from string if will be casted to string before send to VFX for Silverlight project.

The code must be written as a new method of the class cvfpdomainservice. There is an example in cvfpdomainservice class - getdatetime method:

```
LPARAMETERS tcArg

RETURN TRANSFORM(DATETIME())
```

2. Create new method of main COM server class vfpCOMDomainService.

In this case there are needed changes in the VFX for Silverlight VfxDataLayer.Web project, too. It is necessary to add new corresponding methods in VFXDomainService.cs and DataHandler.cs

Additionally, there is GetDateTime method in the COM server, it returns the current date and time that can be used for a COM server test.

Server Site

The COM server at the server side can handle data access to a VFP database. Actually, also other databases, especially SQL Server, can be supported in this way, too. Vfpcomdomainservice.dll uses generated cursor adapter objects for performing VFX for Silverlight service operations.

Data access configuration specified in Config.vfx is used for data source specification.

Data exchange with the VFX for Silverlight project is done in XML format.

Client Site

COM servers are supported at the client site as well. These COM servers can be used only in Silverlight clients running out-of-browser (OOB) with elevated trust. Such COM servers are suitable for any business logic which may run at the client site and for data access.

This technique is especially useful for Silverlight desktop applications. For the COM server the VFP runtime, the COM server itself and a VFP database must be installed. The Silverlight application can handle the whole data access with the COM server.

Localization

Localization in application is managed from several properties of the object *VfxApplicationObjectBase*:

- RuntimeLocalization – determine whether the application will be localized.
- LangID – determine startup language of application (respect when RuntimeLocalization property is set to true). This property contains language abbreviation.

Language	Abbreviation
English	ENG
French	FRE
German	GER
Italian	ITA
Spanish	ESP
Bulgarian	BUL
Greek	GRE
Dutch	NL
Portuguese	POR
Russian	RU
Czech	CZE
Finnish	FIN
Polish	PL
Turkish	TR
Albanian	ALB
Swiss	CHD
Romanian	RO
Slovak	SVK
Estonian	EST
ChineseSimp	CHS
Japanese	JPN
ChineseTrad	CHT

table 1 – list of available languages

- CurrentLanguage – determine current selected language in application (respect when RuntimeLocalization property is set to true).

When application start loading all enabled languages. To enable the application to use a language is necessary to set field *IsActive* in table *VfxLanguage* to true. First selected language is language from property *LangID* of object *VfxApplicationObjectBase* or last selected language for authenticated user.

When language is change load data for culture (decimal separator, date format and other). This information is used to set culture properties. After that application raise event *LangSetupLoad* from class *VfxEventHelper*. Any form listens for the event. The processing of the event is request to load the necessary texts for the given form.

Localization of forms

All form, which support runtime localization implement interface *IVfxLocalization*. In the first occurrence of the event *Loaded* call method *InitializeLocalizationManager(string language)*. In this method will be create instance of object *VfxLocalizationManager*, find all control which support runtime localization (these are controls that have properties *ToolTipMessageID* or *MessageID*).

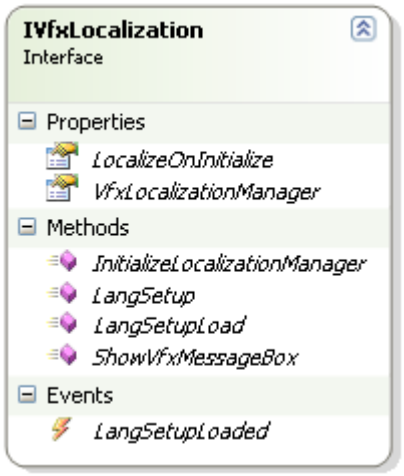


figure 1 – IvfxLocalization interface

Methods, properties and event related to Runtime localization

Properties, methods and events from IvfxLocalization interface:

- LocalizeOnInitialize – determine if search and localize object in form on method *InitializeLocalizationManager*. Default value is true.
- *VfxLocalizationManager* – object of type *VfxLocalizationManagerBase* that help to localize form and objects.
- *InitializeLocalizationManager(string language)* – Initialize *VfxLocalizationManager* property.
- *LangSetup(string language, bool hasError)* – Sets language depending text. This method is call after load data from table vfxmsg.
- *LangSetupLoad(string languge)* – Called when *LangSetupLoad* event was fired. In this method send request to the server to load data from table vfxmsg. Use method *LoadDataFromDB* of class *VfxLocalizationManagerBase*.
- *ShowVfxMessageBox* – this method is used to show localized message. It is possible to send messageID for title and text of the message box. If form is not load relevant constant use default text that can be send as parameters of the methods.
- *LangSetupLoaded* – this event was fired every time after form is localized.

Properties, methods and events provide by form

- *AddAditionalMessageToLocalize()* – in this method developer can add additional constants(user defined or other) that will be loaded and can be used in form. This method is virtual and can be override.

Load/read user defined constant

To use user defined constant in form is necessary to fetch this constant from table vfxmsg. All form has method *AddAditionalMessageToLocalize()*. It is necessary to override thid method and add desired constant. There are two way to add more constant that will be loaded. Use method *AddMessage*. The method accept *string* or *IEnumerable<string>*. Past desired constant as parameters to the method *AddAditionalMessageToLocalize()*.

```
protectedoverridevoid AddAdditionalMessageToLocalize()
        {
base.AddAdditionalMessageToLocalize();
            VfxLocalizationManager.AddMessage("CAP_FRM_MAIL_MERGE");
```

```
List<string> list = newList<string>() { "CAP_FRM_CUSTOMERS",
"MSG_INVALID_DATA", "MSG_ERROR" };
            VfxLocalizationManager.AddMessage(list);
        }
```

The localized message string can be fetched with the method *GetMessageValue(string messageID)* of the instance of VfxLocalizationManagerBase class. Method *GetMessageValue* return localized string or null when string is not found.

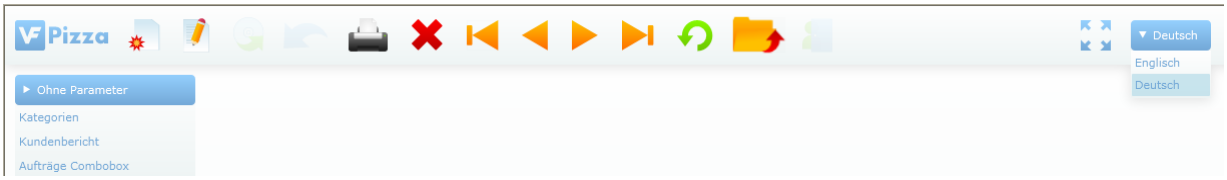
```
string localizedText =
VfxLocalizationManager.GetMessageValue("MSG_INVALID_DATA");
```

Creating Multilingual Applications using VFX for Silverlight

VFX for Silverlight is well prepared for creation of multilingual applications.

Runtime Localization

With VFX for Silverlight it is possible not only to develop localized applications, but also to allow end-users to change the application language at run time.



Runtime localization is implemented in a dynamically and memory saving way in the class *LocalizationManager*.

Runtime localization is controlled by the property VfxApplication.cAppObject.InnerAppObject.RunTimeLocalization. If the value of this property is set to true, the language for the application can be chosen in the login dialog.

Additionally, at run-time, the language can be changed using the language selection combobox, in the main application toolbar.

Each form or other visual object localizes:

- on its visualization;
- on change language from the application toolbar.

Localized controls are all VFX for Silverlight controls in the XAML having *MessageID* or *TooltipMessageID* properties with not empty values. These properties store string values corresponding with the field message_id in the table *Vfxmsg*.

On localization process:

- These values are added to an array <string> object Messages used as QueryParameter of a Domain Context.
- A dictionary object is created and filled. It stores the message_id and the current language text for each value of MessageID or TooltipMessageID of the XAML controls. This object uses caption properties or tooltip properties of localized controls.

The last used language is kept on a per user basis in the table Vfxusr table. Next time when same user logs in the application, the last chosen language will be automatically selected.

Message Localization

To localize a message, form method *LangSetup*must be changed. *Vfxmsg message_id* value must be add to List<string> object Messages before *base.LangSetup()*:

```
protectedoverridevoid LangSetupLoad(string language)
        {
            localizationManager.AddMessage("MSG_PLEASESELECTPRINTER");
base.LangSetupLoad(language);
        }
```

The localized message string can be fetched with the method GetMessageValue(string message_ID) of the instance of LocalizationManager class named localizationManager:

```
localizationManager.GetMessageValue("MSG_PLEASESELECTPRINTER");
```

Display Pictures

There is a VFX for Silverlight control for downloading and displaying picture at client site. This is VfxObject:VfxImage.

1. If there is no `VfxObject` namespace declaration, it has to be added manually in code window at the header part of the XAML:

```
xmlns:VfxObject= "clr-namespace:VfxObject;assembly=VfxObject"
```

If the form is created with VFX for Silverlight wizard, this declaration is generated.

2. Open XAML file in design mode. Find control `VfxObject:VfxImage` in Toolbox and drop in design window.

3.Set VfxSourcePath and VfxSourceFileName properties.

VfxSourcePath defines source path property. In the example below it is “picturepath” property of “Products” entity. It corresponds with field picturepath of Products table in the database.

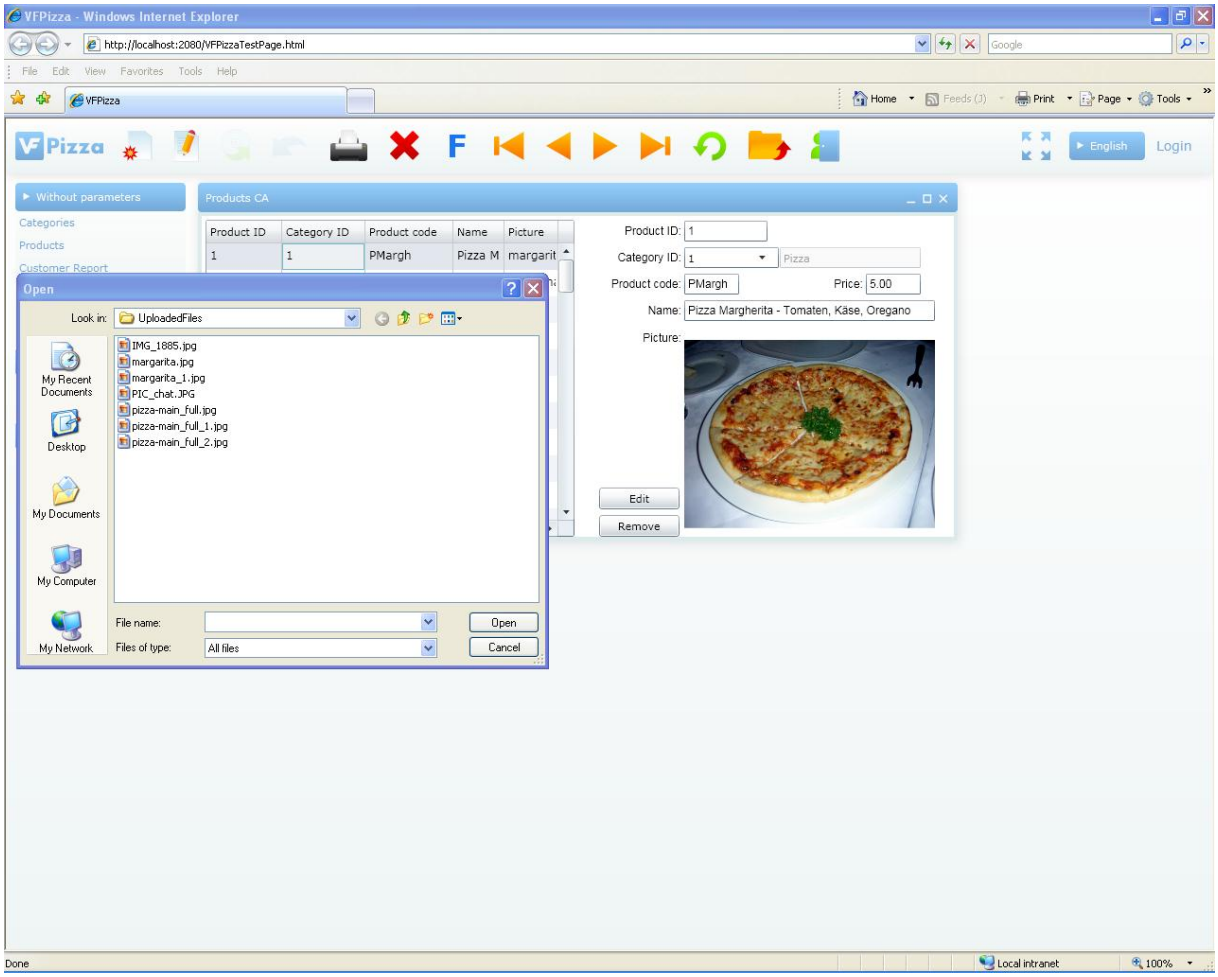
VfxSourceFileName defines file name property. In the example below it is “picturefile” property of “Products entity”.

```
<VfxObject:VfxImage HorizontalAlignment="Left"
MessageID="MSG_PLEASEWAIT" Margin="509,121,0,0" Name="vfxImage1"
Width="265" VfxSourcePath="{Binding
Path=Products.CurrentItem.picturepath}"
 VfxSourceFileName="{Binding Path=Products.CurrentItem.picturefile}"
Height="217" VerticalAlignment="Top" />
```

In addition there can be set MessageID to localize button Content. For more details about localization see the chapter “Run-time Localization”.

Upload and Download files

There are developed classes supporting the upload and download of files, and displaying pictures in VFX for Silverlight. They can be used in easy way in VFX for Silverlight forms.



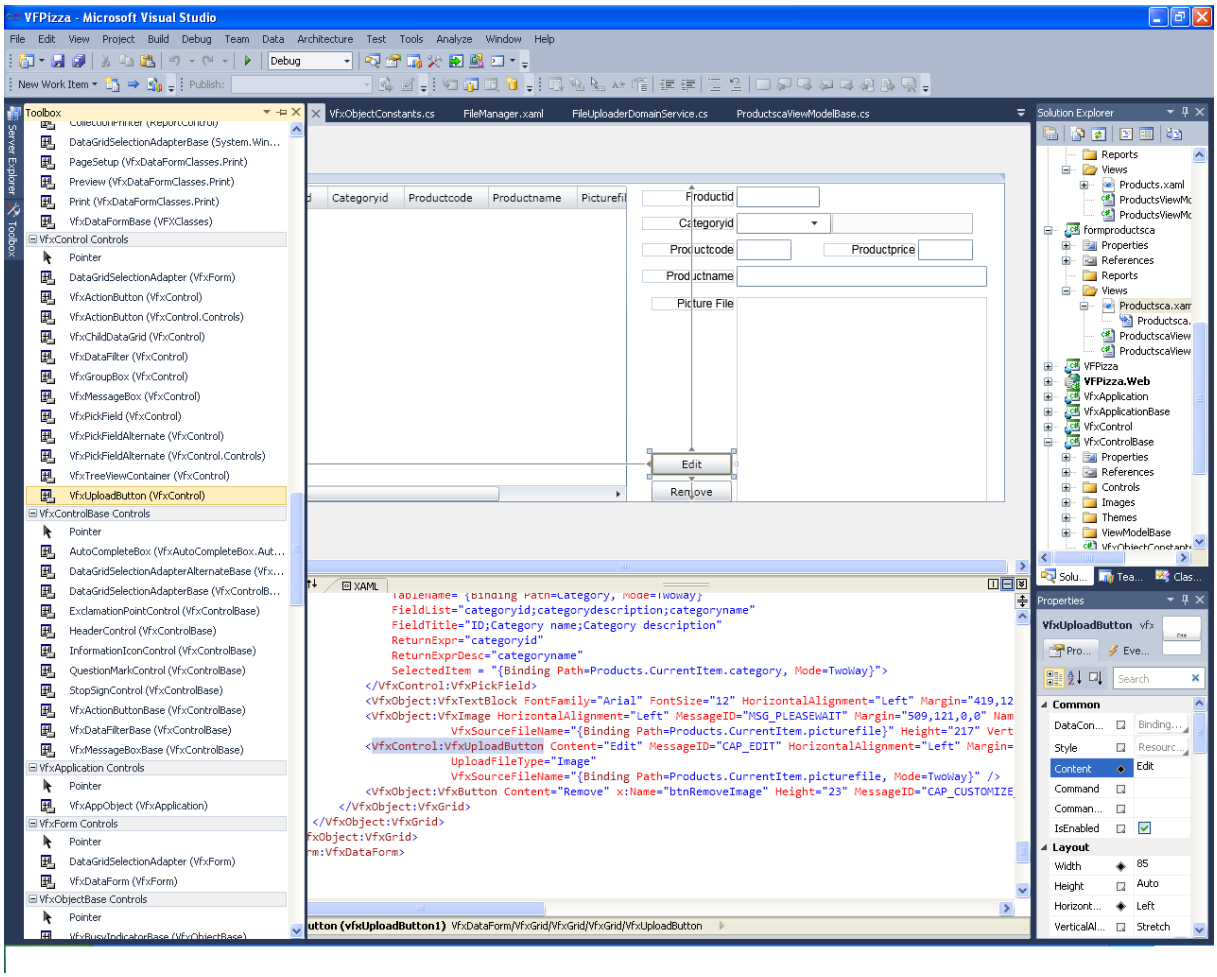
Upload Files

If there is no VfxControl namespace declaration, it has to be added manually in code window at the header part of the XAML:

```
xmlns:VfxControl= "clr-namespace:VfxControl;assembly=VfxControl"
```

If the form is created with the VFX for Silverlight Wizard, this declaration is generated.

Open XAML file in design mode. Find control VfxControl:VfxUploadButton in Toolbox and drop in design window.



There will be generated the control declaration in XAML window:

```
<VfxControl:VfxUploadButton Content="Button" Height="23"
HorizontalAlignment="Left" Margin="292,298,0,0" Name="vfxUploadButton1"
VerticalAlignment="Top" Width="75" />
```

VfxUploadButton have functionality to upload selected at client site picture, upload it in server folder. If the control is bind after form saving file path and file name will be saved in the database.

Set UploadFileType, VfxSourcePath and VfxSourceFileName properties.

UploadFileType defines file type filter for open file dialog on file uploading. It accepts two values “All” and “Image”. Default UploadFileType value is “All”. If Images will be uploaded it is recommended to set this property to “Image”.

```
UploadFileType="Image"
```

Other way to filter displayed files in open file dialog is using the property OpenFileDialogFilter. For example:

```
OpenFileDialogFilter="Documents|*.doc;*.txt|Images|*.jpeg;*.jpg;*.png"
```

VfxSourcePath defines source path property. In the example below it is “picturepath” property of “Products” entity. It corresponds with field picturepath of Products table in the database.

VfxSourceFileName defines file name property. In the example below it is “picturefile” property of “Products entity”.

```
<VfxControl:VfxUploadButton Content="Edit" MessageID="CAP_EDIT"
HorizontalAlignment="Left" Margin="419,286,0,29" Name="vfxUploadButton1"
Width="85"
 UploadFileType="Image"
 VfxSourcePath="{Binding Path=Products.CurrentItem.picturepath,
Mode=TwoWay}"
```

```
 VfxSourceFileName="{Binding Path=Products.CurrentItem.picturefile,
Mode=TwoWay}" />
```

In addition there can be set MessageID to localize button Content. For more details about localization see the chapter “Run-time Localization”.

Data Handling

Data handling is based on the Model-View-ViewModel (MVVM) pattern, Entity Data Model and WCF RIA Services technologies.

All applications work with SQL server databases and VFP databases. Common model and services are used in both cases.

The database type is set in the property CurrentDomainContext of the application object. Default is SQL server database.

All base functionality needed for data handling is included in the VFX for Silverlight template project. All application specific functionality is generated by the VFX – Silverlight Wizard.

Error tracking

When an error is raised, there is an exception thrown in Silverlight. It will be handled and will be tracked automatically in a table. The name of the current user, date, time, error message, as well as the stack trace, is saved. Further application behavior after an exception was thrown can be set through properties of the application object.

The exception information is logged in Vfxlog table.

- The content of the information depends on ErrorDetailLevel application property. Itaccepts the following values:
    - MessageOnly – Only message property of exception object will be logged.
    - NoCallStackInformation – All information except for Call Stack Information will be logged.
    - FullDetailedInformation - All information will be logged:

  Message;

  StackTrace;

  EntitySets list;

  Exception.Data - additional user-defined information about the exception.

  The name of the current user, date and time will be logged in all cases.

There is a different behavior depending on application mode:

- Debug mode

Fully exception information will be shown in a message.

- Release mode
    - Exception type will be checked.

      If it is one of types defined in the application property CriticalErrorTypes, a user friendly dialog is shown.

      If the error type is not defined in CriticalErrorTypes application property, the error will be ignored and execution will continue.
    - A user friendly dialog will be shown if exception log failed.

Features for Developers

VFX – Resource Table

VFX applications use a resource table to store all information about forms the user has opened since the last initialization. This information is used to set the size and position of the Form, DataGrid layouts, as well as the current sort order.

Here are the settings stored on a per user base in the VFX resource table.

Position and size of the form are saved on form close. The user sees the forms always appearing exactly the same way he closed it.

You can reset this VFX Resource File by clicking the Clear Resource button in the user list form. This will delete all entries from the VFX Resource File.

Tips

1. If an application does not start in the browser, check StartUp project and start page.

<application name>.Web project must be set as Start Up project. VFXforSilverlight.html must be set as start page.

2. To debug in Silverlight .cs files, Silverlight debugger has to be enabled. If it is not:

Find main project <application name>.web, open context menu and choose “Properties”. At “Properties” window choose “Web” tab. At the bottom of the tab check “Silverlight”.