AGILEDIALOGS SDK (BETA)

Table of Contents

AgileDialogs SDK (BETA)

1. Introduction

This document describes the mechanisms that AgileDialogs provides to include custom controls in dialogs pages.

These custom controls are developed in Silverlight (Version 4).

1.1 Disclaimer of warranty

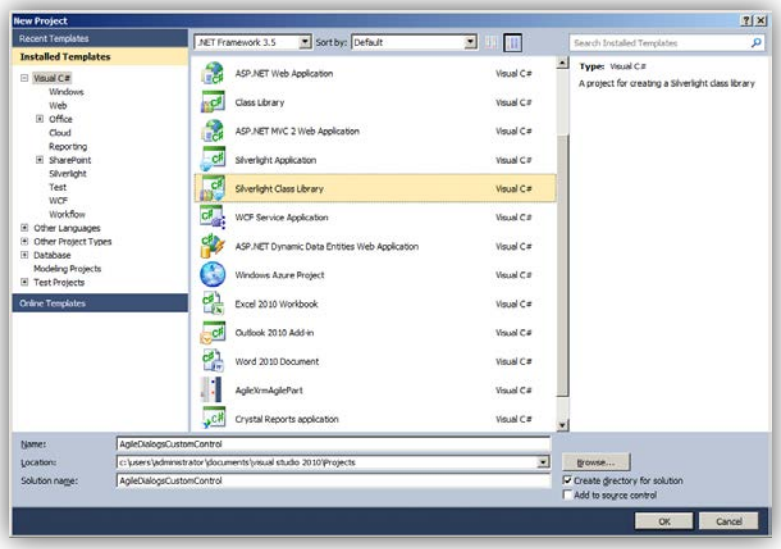
AgilePoint Inc. makes no representations or warranties, either express or implied, by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

2. Developing a custom control

Custom controls for AgileDialogs are developed in Silverlight 4 using Visual Studio. AgileDialogs provides an interface that this custom controls should implement if the control is used to set values in dialog context.

2.1 Creating the project

To develop a custom control create a new Silverlight Class Library Project in Visual Studio:



2.1.1 Required References

In order to develop the custom controls the following references must be added:

- AgileDialogsSDK
- AgileDialogsSilverlightControls

If other references are added to the project the DLL's must be deployed with the DLL that contains the custom control. For instance, if third party controls are used the DLLs of these controls must be deployed in order to make custom control work.

2.1.2 Coding the custom control

AgileDialogs controls get information from the user and put this information in the Dialog context to use the values gathered from the user to take actions in further steps or dialogs automatic activities.
To set values in context and provide validation at runtime the custom control must implement the interface: `AgileDialogsSilverlightControls.Common.IAgileScriptsControl`.
This interface has the following definition:

```
public interface IAgileScriptsControl
{
   string GetValue();
   string GetDisplayValue();
   bool isValid();
   bool Required{get;set;}
   string GetValueVariableName();
   string GetDisplayVariableName();
   void SetDefaultValue(string defaultValue);
}
```

GetValue Method

This method returns a string with the value selected or set by the user in the custom control. This is the value that will be stored in dialog context in a variable with the name returned by the method `GetValueVariableName`.

GetDisplayValue Method

If the control returns a value to be presented to the user that is different from the internal value (for instance, a date is internally stored as a string in sortable format and is presented to the user using client locale) this method should return the readable value, otherwise return the same value that GetValue.
This is the value that will be stored in dialog context in a variable with the name returned by the method `GetDisplayVariableName`.

isValid Method

The dialog cannot continue until all controls return is valid true, so if the value is not valid this method should return false.

Required Property

When this property is true the user must set or select a value before the validation is passed.

GetValueVariableName Method

This is the name of the context variable where the value set by the user will be stored.
Usually this value should return the name of the control so the recommended coding is:

```
public string GetValueVariableName()
{
   return this.Name;
}
```

GetDisplayVariableName Method

This is the name of the context variable where the display value set by the user will be stored.
Usually this value should return the name of the control plus “d” (this is the default value when configuring dialogs in Envision) so the recommended coding is:

```
public string GetDisplayVariableName()
{
  return this.Name + “d”;
}
```

SetDefaultValue Method

This method is called by AgileDialogs Runtime Engine when the user has set a default value in the configuration.
This method should manage the assignation of this default value taken into account the internal behavior of the control.

2.1.3 Sample Code

This is the sample code for a *Hello World* control, with a text block to show a message:

```
public partial class MyDialogsCustomControl : UserControl, AgileDialogsSilverlightControls.Common.IAgileScriptsControl
{
        string _myValue;
        public MyDialogsCustomControl()
        {
            InitializeComponent();
        }

        public string GetDisplayValue()
        {
            return "Hello World!!!";
        }

        public string GetDisplayVariableName()
        {
            return this.Name + "d";
        }

        public string GetValue()
        {
            return myTextBlock.Text;
        }

        public string GetValueVariableName()
        {
            return this.Name;
        }

        public bool Required
        {
            get
            {
                return false;
            }
            set
            {
            }
        }

        public void SetDefaultValue(string defaultValue)
        {
            _myValue = defaultValue;
            myTextBlock.Text = defaultValue;
        }

        public bool isValid()
        {
            return true;
        }
}
```

3. Deploying a Custom Control

There are two types of custom controls:

- Controls that do not have references to third party DLLs
- Controls that have references to third party DLLs

The DLLs with Controls that do not need other DLLs must be copied to the directory *CustomDlls* located in the directory where AgileDialogs web application is installed, for instance *C:\AgileXRM\AgileDialogs\CustomDlls*.
When the DLL has references, the custom control DLL and referenced DLLs must be included in a ZIP file and copied to *CustomDlls* directory.
AgileDialogs Runtime Engine loads custom controls dynamically from the directory *CustomDlls*.