# App Configurations for Branding

## Contents

## Introduction

The sample project created in this document can be downloaded from http://www.limnor.com/studio/AppWithBranding.zip.

Application Configuration can be used to remember user preferences, as shown in http://www.limnor.com/support/UseAppConfig.pdf. It can also be used to give your software branding. We'll use a sample project to show how to weave branding features into your software.

## Programming Tasks

Using application configurations in your software involves following programming tasks.

1. Determine what you want to allow to be changed at runtime; let's call such values "**Software Values**". For example, menu captions, window sizes, text and images on forms, etc.
2. Add Application Configuration components to your software.
3. For every Software Value in your software create a corresponding configuration definition in an Application Configuration component. Let's call such values "**Configuration Values**". So, for every piece of "Software Value", you create a corresponding "Configuration Value" in an Application Configuration component. One pair of "Software Value" and "Configuration Value" forms one configuration in your software.

4. For every configuration in your software, create one action to pass corresponding “Configuration Value” to “Software Value”.
5. Identify events of the first time “Software Values” might be used, for example, when a form is loaded. Assign actions created in the above programming task 4 to such events.
6. For software branding, provide a command (i.e. a menu item) to allow the user to modify “Configuration Values”. Your users may create different sets of “Configuration Values”. Your users may create “named set” of “Configuration Values”; each set of “Configuration Values” is identified by a name. A user may create user-specific set of “Configuration Values”, which is specific to current computer user. Each set of “Configuration Values” represents one brand.

We’ll use a sample project to show how each programming task is accomplished.

Task 6 can also be in a separate program. See
http://www.limnor.com/support/CreateAppConfigUtility.pdf.

## Determine and Use Software Values

This task is to link your business requirements with your software design.

### Determine Software Values

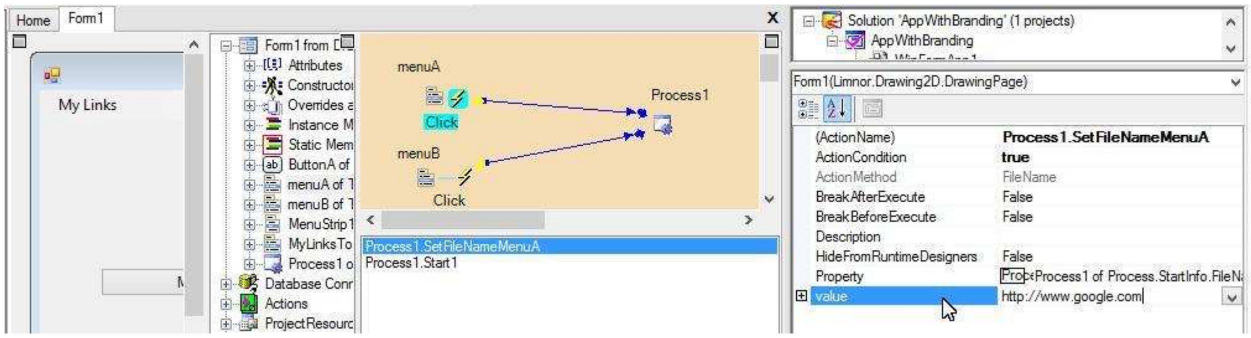
Suppose in our software, we have a form and some menus. We want some menu items to launch web sites in the default web browser. We want the user to be able to set the menu item caption and the web site URL. So, for each such a menu item, we have two Software Values: menu item caption text and the web URL. We may create a new property of the form to hold the web URL. Thus, for each such a menu item, we have two Software Values: Text property of the menu item and the new property we created to hold a web URL.

Suppose we have a button; clicking the button will launch another form. The new form has a web browser control on it. Suppose we want the user to be able to set the button caption and the web URL to be used by the web browser control on the form. We may create a new property of the form to hold the web URL. So, for this button, we have two Software Values: Text property of the button and the new property we created to hold a web URL.

In summary, a “Software Value” is a writeable property of a component. It must be writable because we want to allow the users to set it at runtime. If in your programming you use a constant value then you must create a new writable property to replace the constant value so that the users may modify it.

### Create property to replace constant

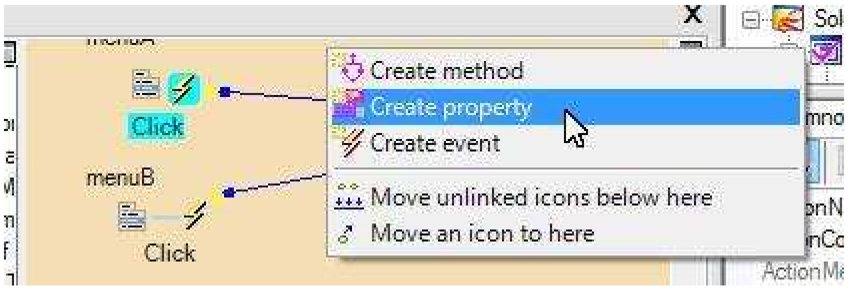
Without weaving branding into our software, we use constant web URL in our actions.



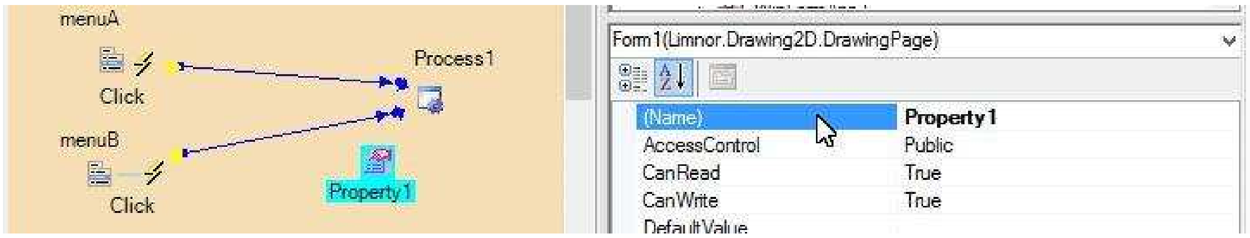
| | |
|---|---|
| (ActionName) | **Process1.SetFileNameMenuA** |
| ActionCondition | **true** |
| ActionMethod | FileName |
| BreakAfterExecute | False |
| BreakBeforeExecute | False |
| Description | |
| HideFromRuntimeDesigners | False |
| Property | ProcProcess1 of Process.StartInfo.FileN |
| value | http://www.google.com |

menuA launches http://www.google.com. For our branding requirement, we need to create a new property to hold a web URL and use the new property in the action.

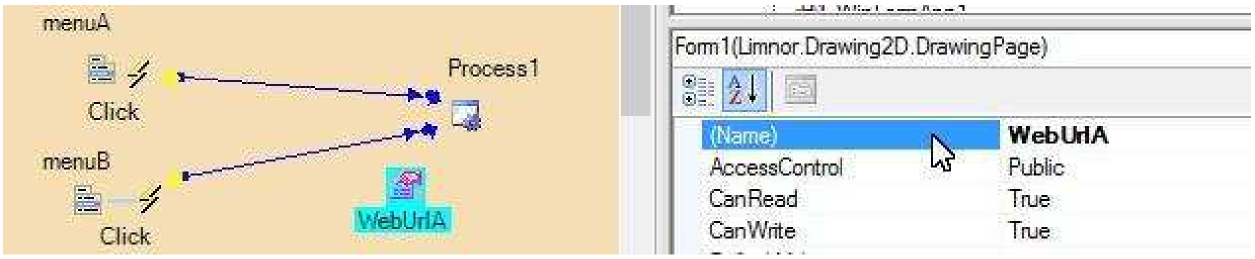
## Create new property



Rename the new property to WebUrlA:

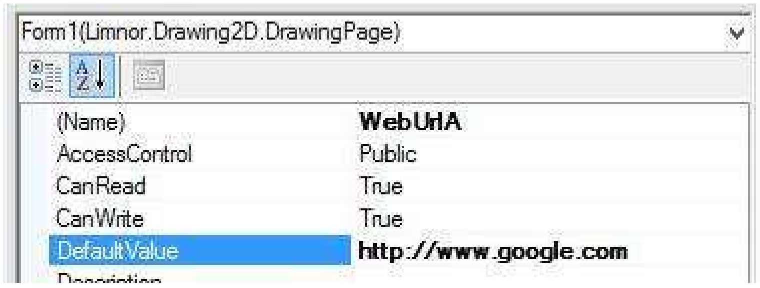


| | |
|---|---|
| (Name) | **Property1** |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |
| DefaultValue | |



| | |
|---|---|
| (Name) | **WebUrlA** |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |

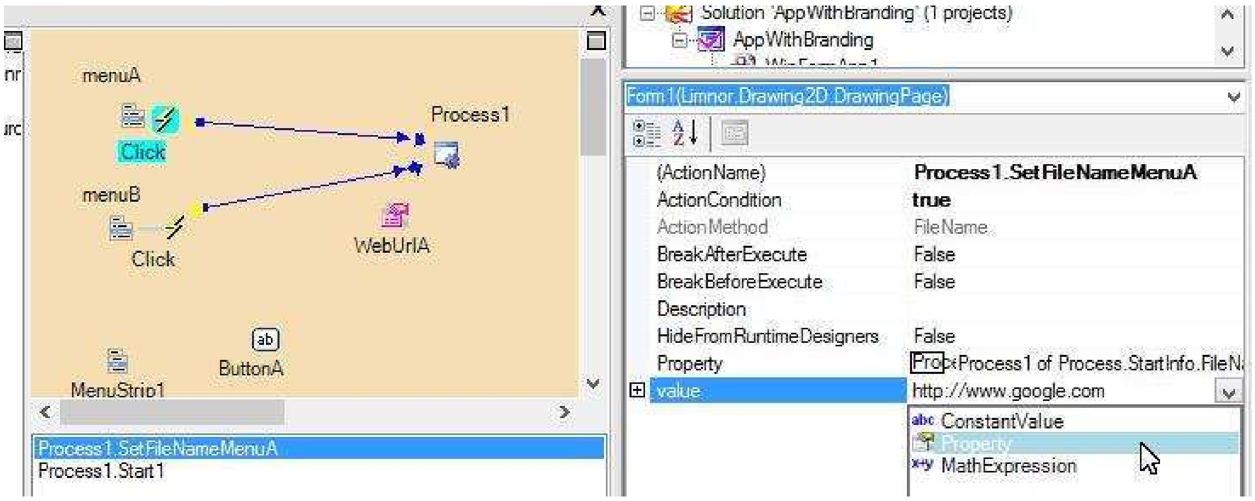
Set its default value to http://www.google.com.



| | |
|---|---|
| (Name) | WebUrlA |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |
| DefaultValue | http://www.google.com |

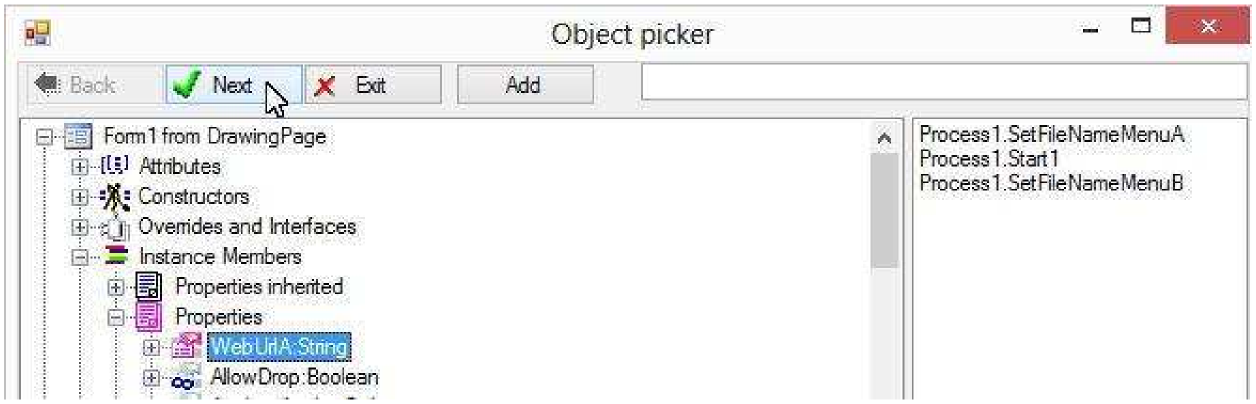
## Replace constant with new property

Use the new property in the action:

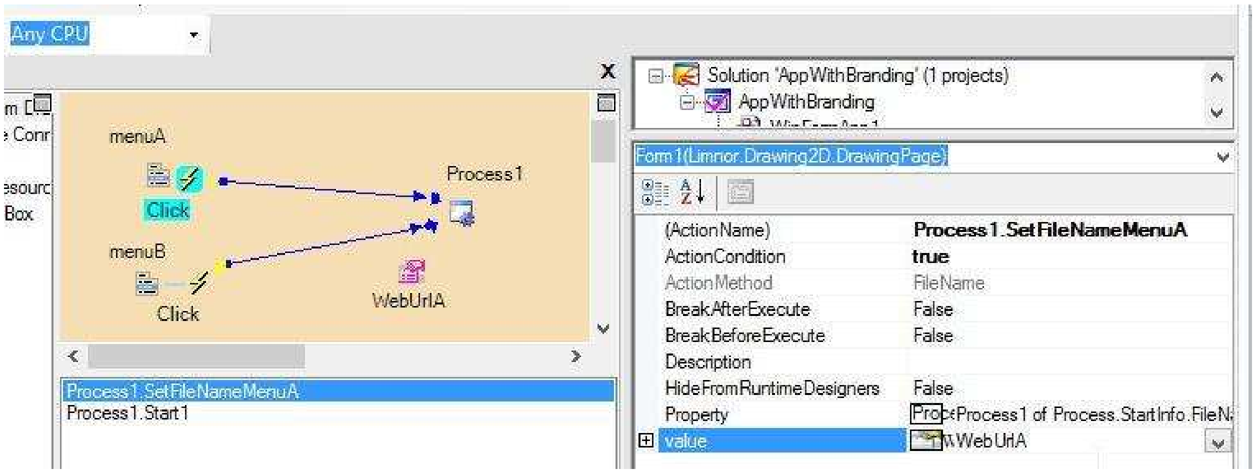


| | |
|---|---|
| (ActionName) | Process1.SetFileNameMenuA |
| ActionCondition | true |
| ActionMethod | FileName |
| BreakAfterExecute | False |
| BreakBeforeExecute | False |
| Description | |
| HideFromRuntimeDesigners | False |
| Property | Process1 of Process.StartInfo.FileN |
| value | http://www.google.com |

Select the new property:



Now the action is using the new property for the web URL, instead of a constant value:

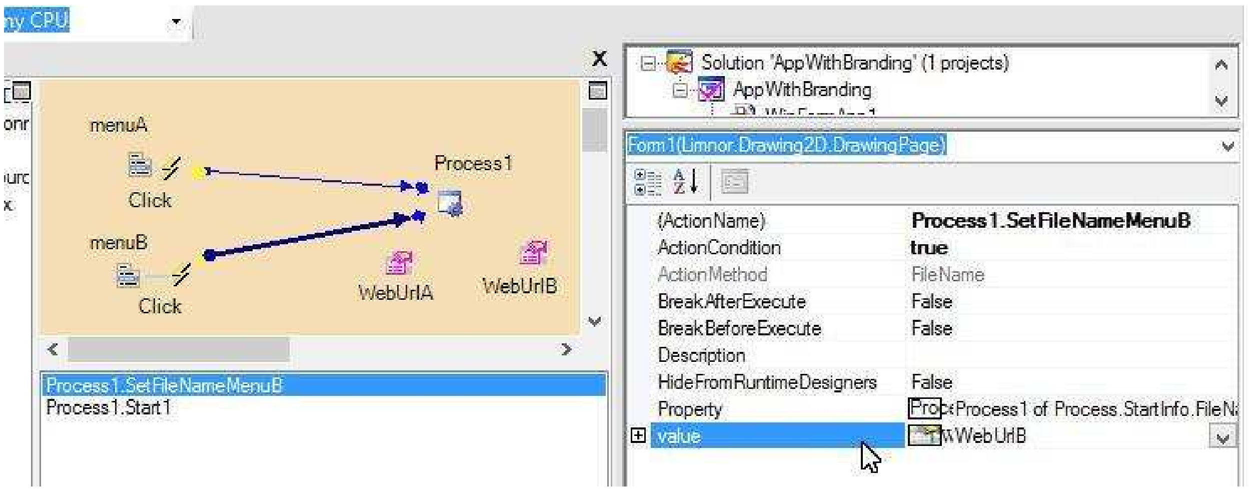

Any CPU
menuA
Click
menuB
Click
Process1
WebUrlA
Solution 'AppWithBranding' (1 projects)
AppWithBranding
Form1(Limnor.Drawing2D.DrawingPage)

Process1.SetFileNameMenuA
Process1.Start1

| | |
|---|---|
| (ActionName) | Process1.SetFileNameMenuA |
| ActionCondition | true |
| ActionMethod | FileName |
| BreakAfterExecute | False |
| BreakBeforeExecute | False |
| Description | |
| HideFromRuntimeDesigners | False |
| Property | Proc Process1 of Process.StartInfo.FileN |
| value | WebUrlA |

Thus, the user may change which web site to launch.

We may do the same for menuB: create a new property, say, WebUrlB; set its default value to http://www.yahoo.com; replace the constant value of http://www.yahoo.com to this new property in our original programming:


Any CPU
menuA
Click
menuB
Click
Process1
WebUrlA
WebUrlB
Solution 'AppWithBranding' (1 projects)
AppWithBranding
Form1(Limnor.Drawing2D.DrawingPage)

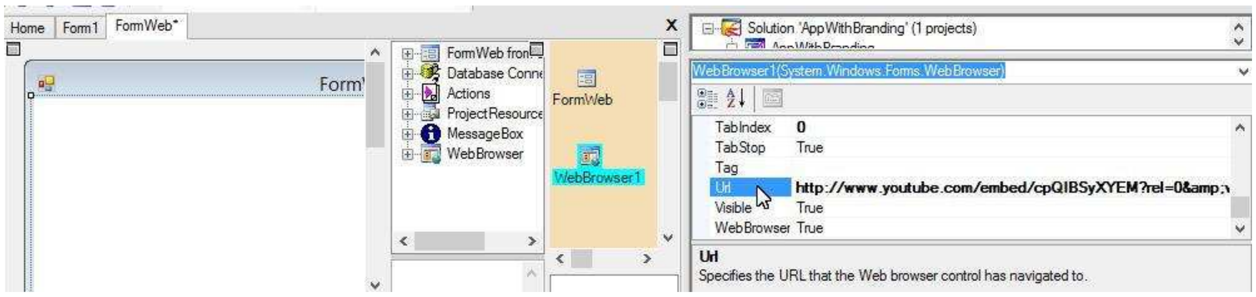
Process1.SetFileNameMenuB
Process1.Start1

| | |
|---|---|
| (ActionName) | Process1.SetFileNameMenuB |
| ActionCondition | true |
| ActionMethod | FileName |
| BreakAfterExecute | False |
| BreakBeforeExecute | False |
| Description | |
| HideFromRuntimeDesigners | False |
| Property | Proc Process1 of Process.StartInfo.FileN |
| value | WebUrlB |

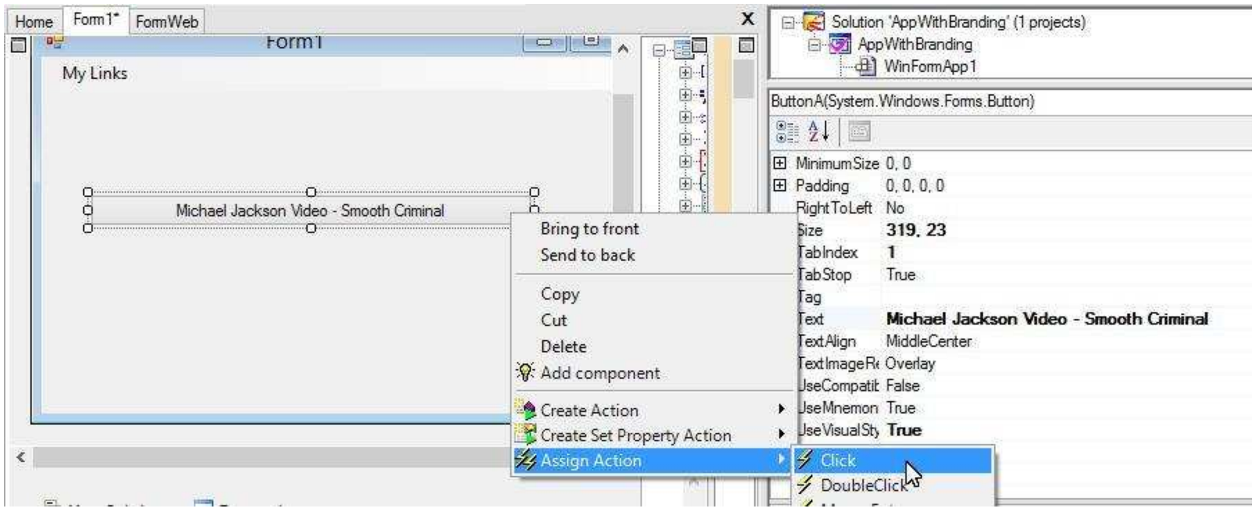
## Software Values across forms

### Programming with default form instance

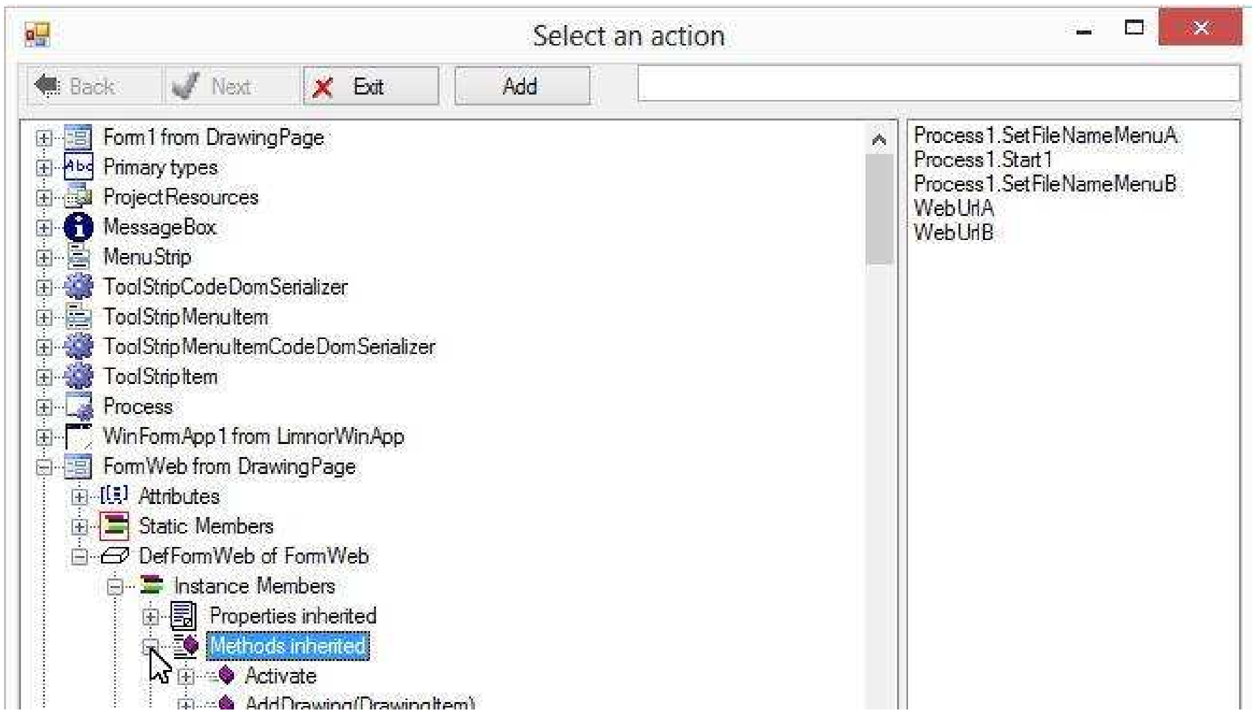
Suppose we have a form containing a web browser control. The Url of the web control is set at the design time:

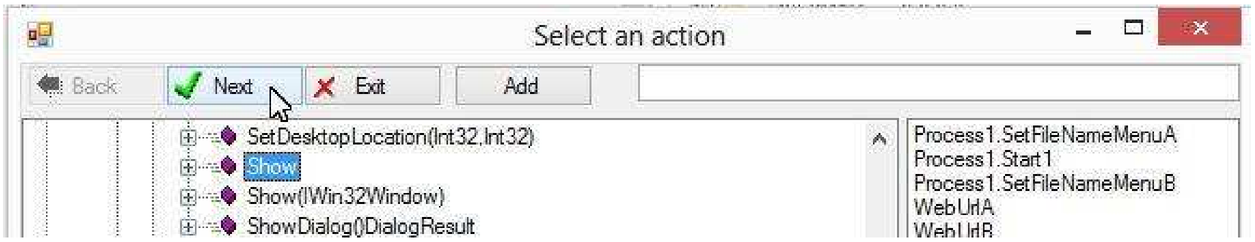

Home
Form1
FormWeb*
FormWeb
WebBrowser1
Solution 'AppWithBranding' (1 projects)
WebBrowser1(System.Windows.Forms.WebBrowser)
TabIndex 0
TabStop True
Tag
Url http://www.youtube.com/embed/cpQIBSyXYEM?rel=0&amp;
Visible True
WebBrowser True
Url
Specifies the URL that the Web browser control has navigated to.

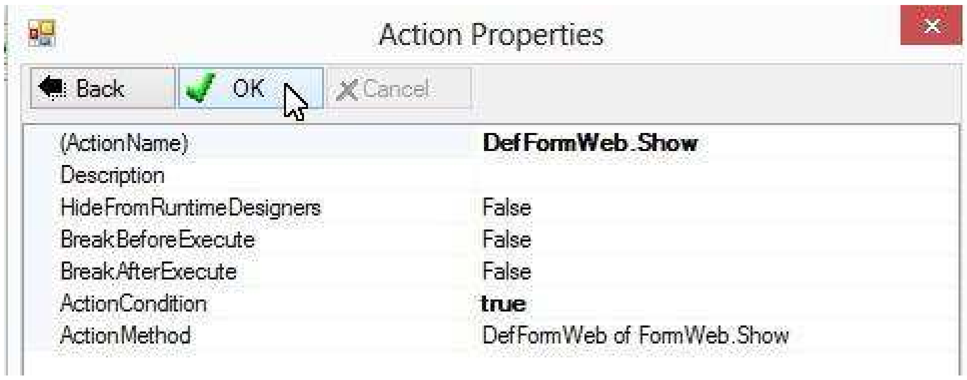
We have a button in our main form to show this web browser form, using default form instance. For details of using default form instance, see http://www.limnor.com/support/UseDefaultFormInstance.pdf. For your convenience, we show such programming below. Right-click the button; choose "Assign Action"; choose "Click" event:



Choose Show method of the default instance of the web browser form:

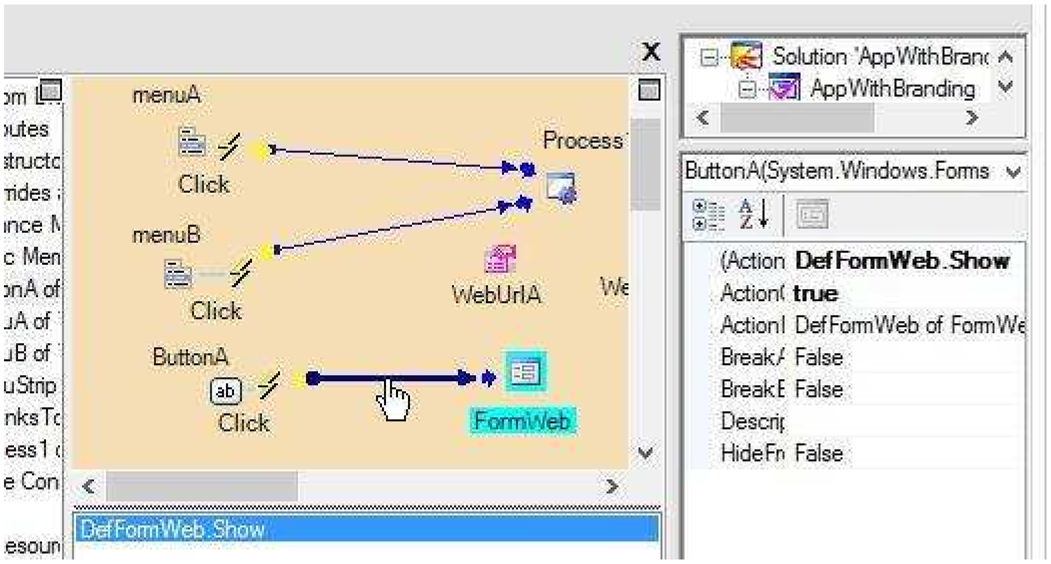






| | |
|---|---|
| (ActionName) | **DefFormWeb.Show** |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | **true** |
| ActionMethod | DefFormWeb of FormWeb.Show |

The action is created and assigned to the button:

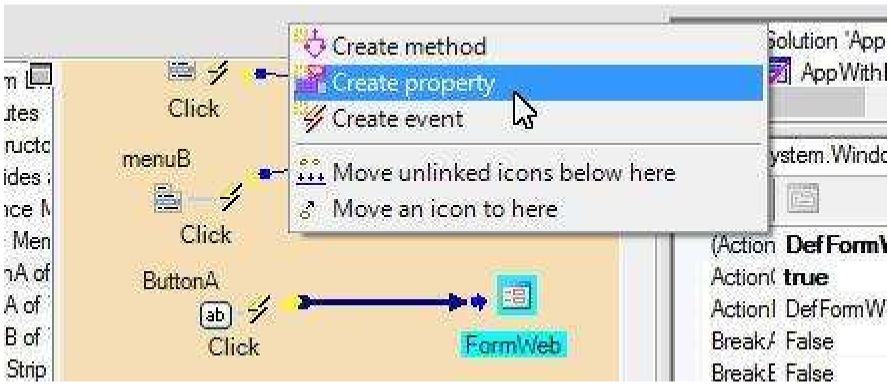


| | |
|---|---|
| (Action | DefFormWeb.Show |
| Action( | true |
| ActionI | DefFormWeb of FormWe |
| Break/ | False |
| BreakE | False |
| Descrip | |
| HideFr | False |

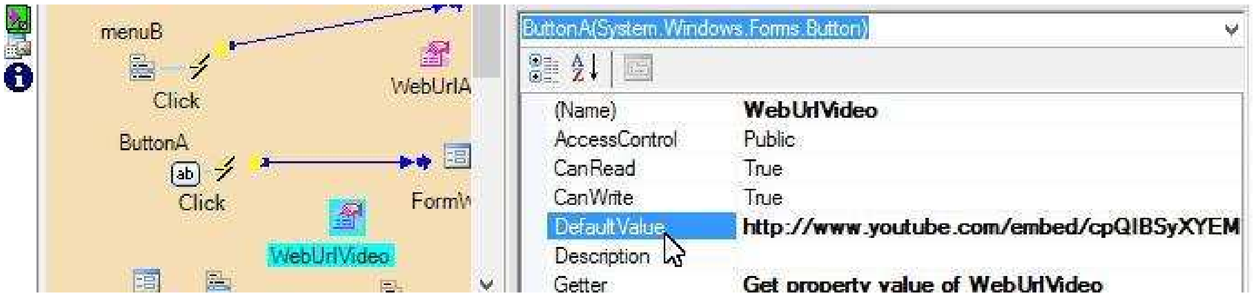
## Software Values

Suppose we want to allow the users to set the button caption and the web URL for the web browser control. So, we have two Software Values: the Text property of the button and the web URL.

Create a new property for web URL used by the button:



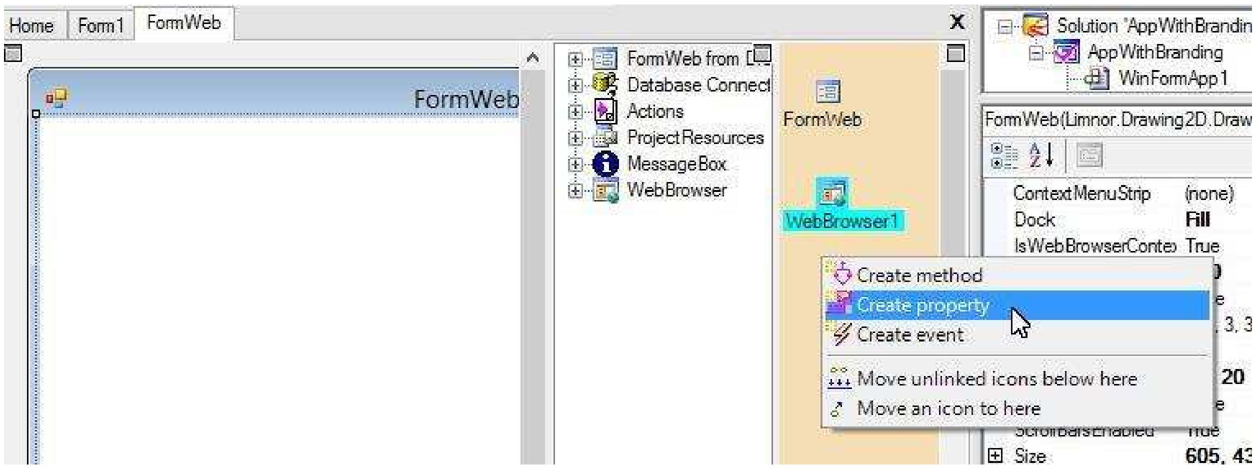
Rename the property to WebUrlVideo and set its default value to the original design time setting:



| | |
|---|---|
| (Name) | WebUrlVideo |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |
| DefaultValue | http://www.youtube.com/embed/cpQIBSyXYEM |
| Description | |
| Getter | Get property value of WebUrlVideo |

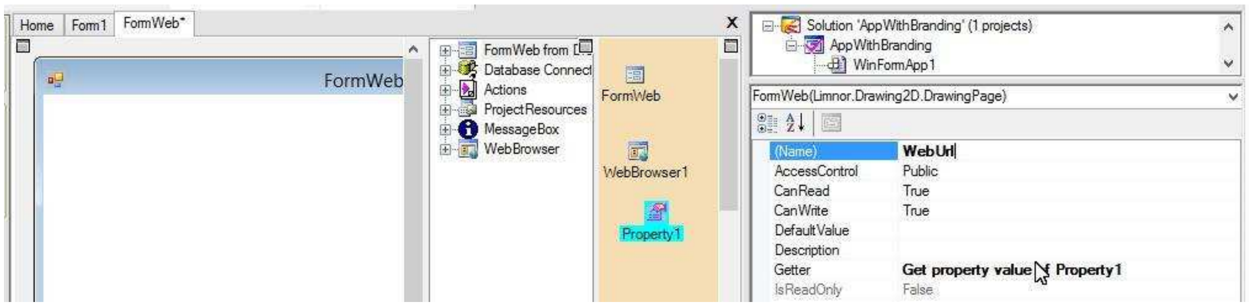
## Passing data across forms

When the user clicks the button, we want to set the Url property of the web browser control. But the web browser control is on another form, thus the Url property is not in the same scope of the button. That is why you cannot see it under the default instance of the web browser form. We may create a property of the web browser form to make the internal Url property accessible from outside.



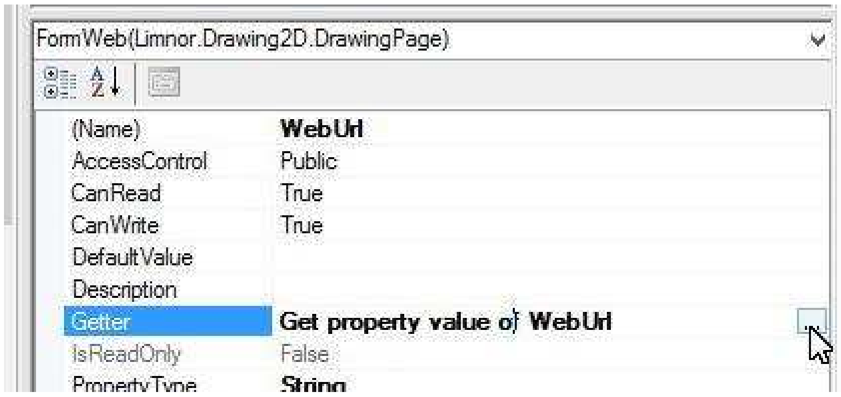
| | |
|---|---|
| ContextMenuStrip | (none) |
| Dock | Fill |
| IsWebBrowserConte | True |
| Size | 605, 43 |

Rename the new property to WebUrl:



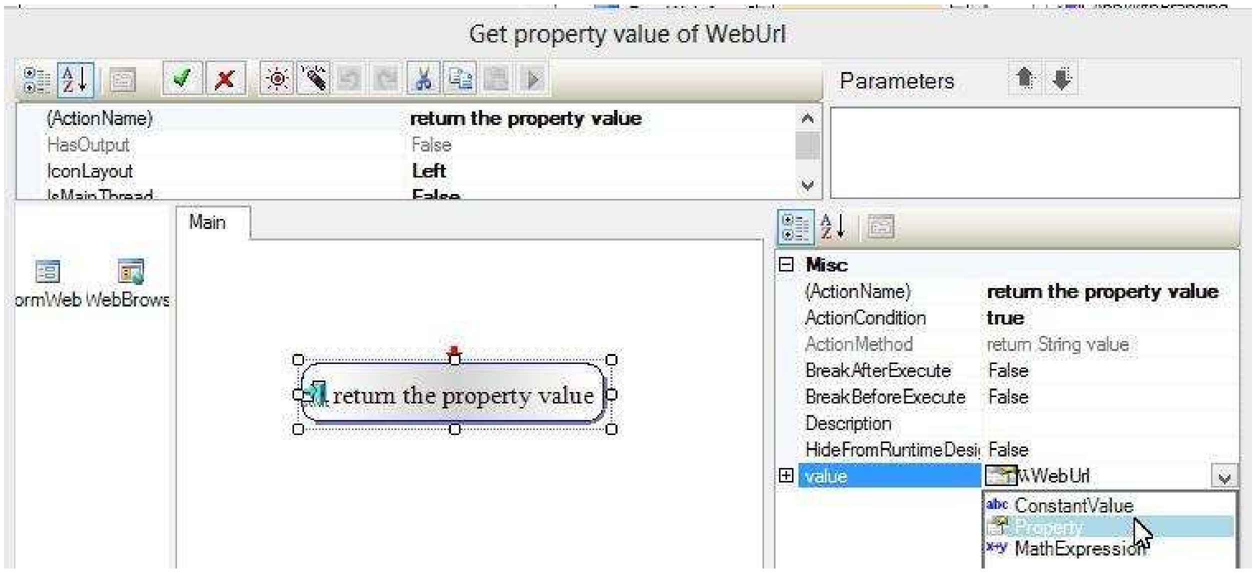
| | |
|---|---|
| (Name) | WebUrl |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |
| DefaultValue | |
| Description | |
| Getter | Get property value of Property1 |
| IsReadOnly | False |

Set the Getter of the new property to make the new property returns the Url property of the web browser control:



| | |
|---|---|
| (Name) | WebUrl |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |
| DefaultValue | |
| Description | |
| Getter | Get property value of WebUrl |
| IsReadOnly | False |
| PropertyType | String |

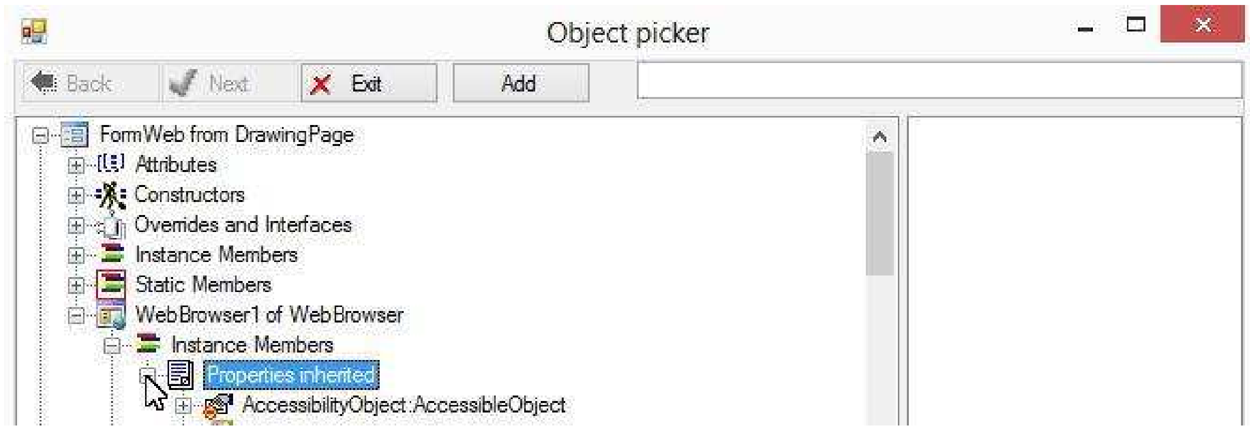
By default, the Getter contains one action to return an internal value for WebUrl property. We modify the return value to make it return the Url property of the web browser:

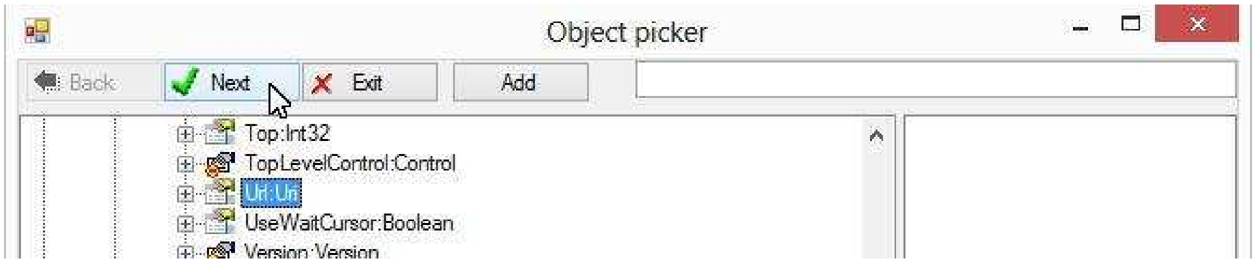


| | |
|---|---|
| (ActionName) | return the property value |
| HasOutput | False |
| IconLayout | Left |

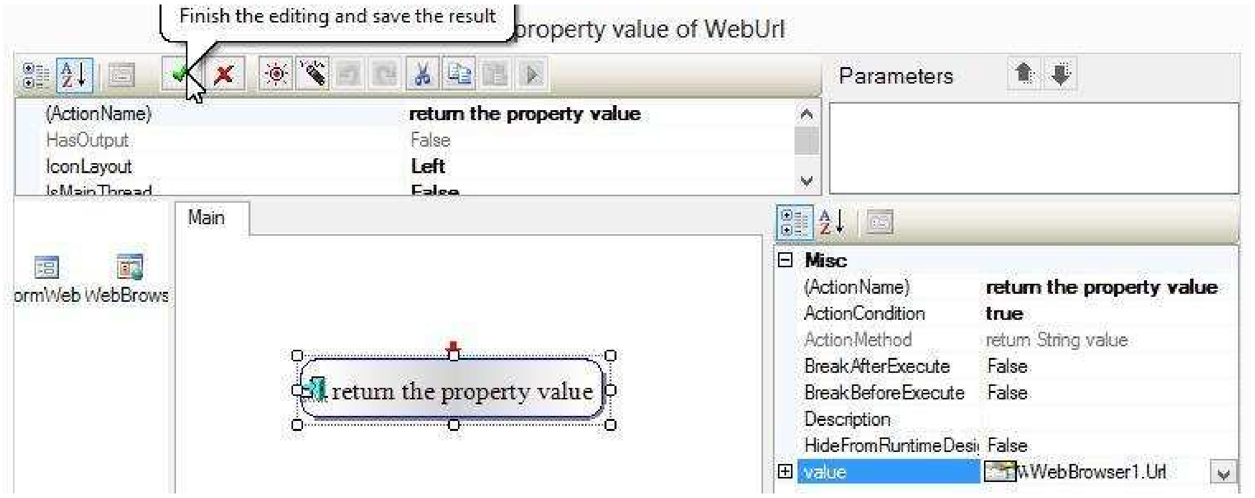
Misc

| | |
|---|---|
| (ActionName) | return the property value |
| ActionCondition | true |
| ActionMethod | return String value |
| BreakAfterExecute | False |
| BreakBeforeExecute | False |
| Description | |
| HideFromRuntimeDesi | False |
| value | WebUrl |





Finish editing the Getter:


Finish the editing and save the result
property value of WebUrl
Parameters
Main
ormWeb WebBrows
return the property value

| | |
|---|---|
| (ActionName) | return the property value |
| HasOutput | False |
| IconLayout | Left |
| IsMainThread | False |

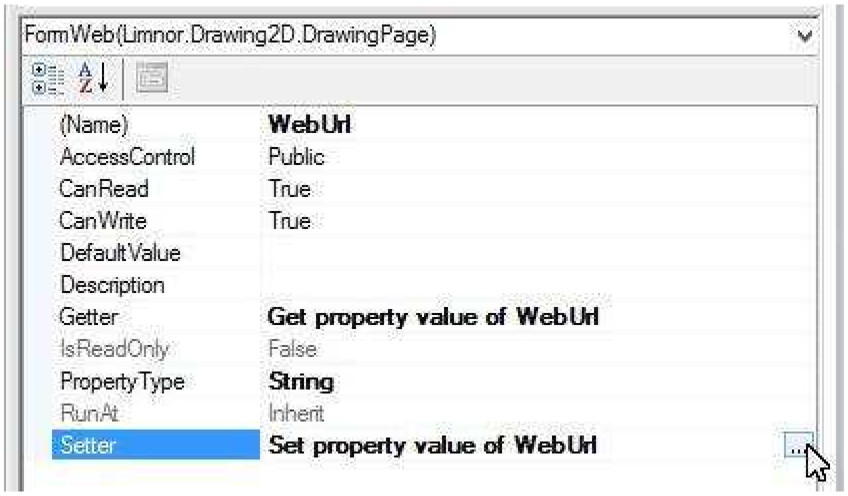
Misc

| | |
|---|---|
| (ActionName) | return the property value |
| ActionCondition | true |
| ActionMethod | return String value |
| BreakAfterExecute | False |
| BreakBeforeExecute | False |
| Description | |
| HideFromRuntimeDesi | False |
| value | WebBrowser1.Url |

Now the WebUrl property will return the same value of the Url property of the web browser control.

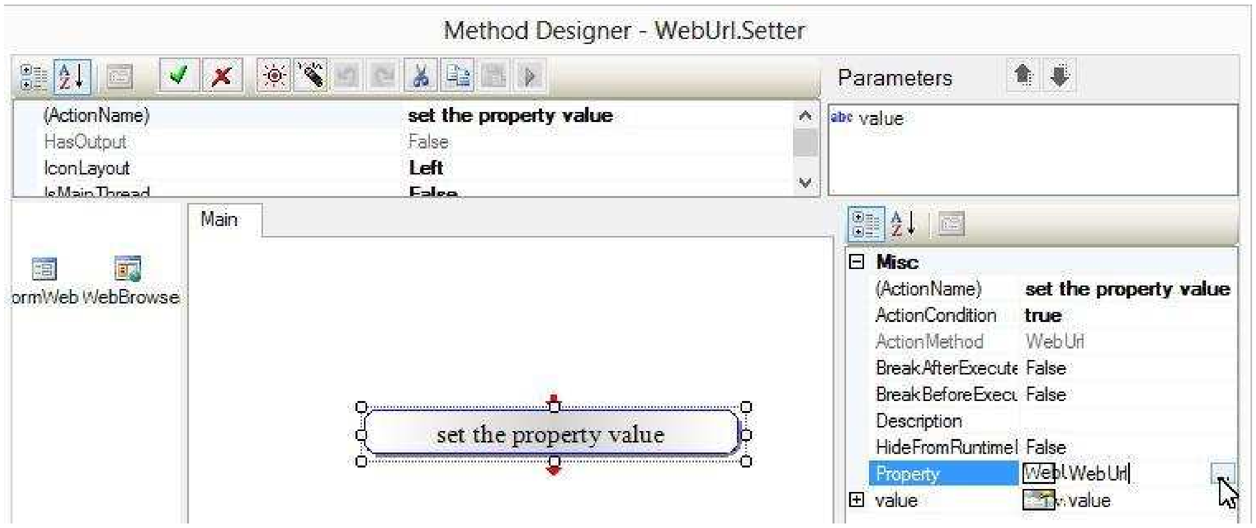
A property has a Setter which is executed for setting the property value.

Edit the Setter of the new property to set the Url property of the web browser control:


FormWeb(Limnor.Drawing2D.DrawingPage)

| | |
|---|---|
| (Name) | WebUrl |
| AccessControl | Public |
| CanRead | True |
| CanWrite | True |
| DefaultValue | |
| Description | |
| Getter | Get property value of WebUrl |
| IsReadOnly | False |
| PropertyType | String |
| RunAt | Inherit |
| Setter | Set property value of WebUrl |

By default, the Setter contains one action to set the value of the internal value of the property. We may modify it to make it set the Url property of the web browser control:

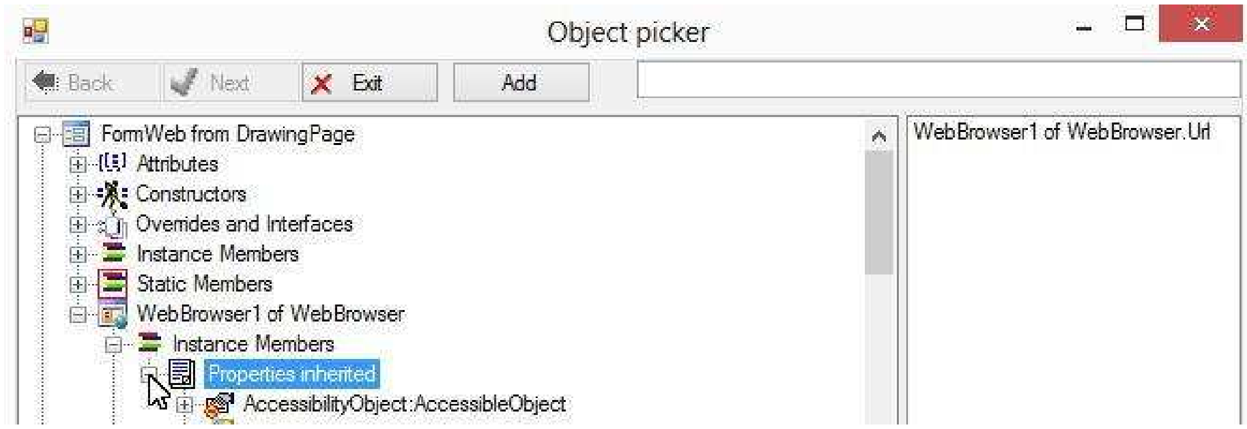


| | |
|---|---|
| (ActionName) | set the property value |
| HasOutput | False |
| IconLayout | Left |
| IsMainThread | False |

abc value

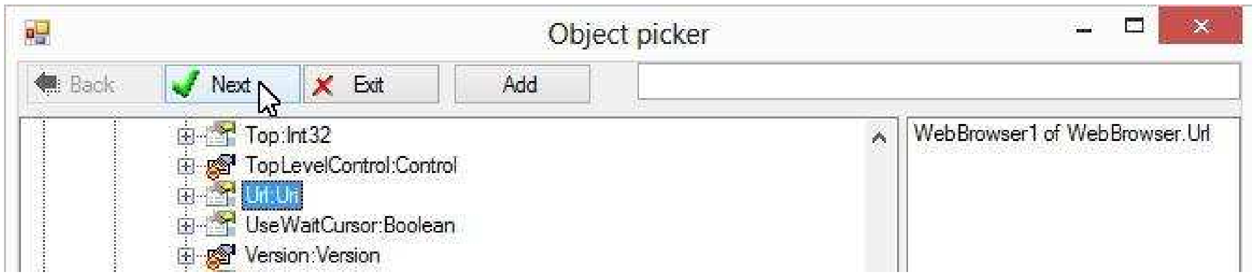
Misc

| | |
|---|---|
| (ActionName) | set the property value |
| ActionCondition | true |
| ActionMethod | WebUrl |
| BreakAfterExecute | False |
| BreakBeforeExecu | False |
| Description | |
| HideFromRuntime | False |
| Property | WebUrl.WebUrl |
| value | v.value |



FormWeb from DrawingPage
Attributes
Constructors
Overrides and Interfaces
Instance Members
Static Members
WebBrowser1 of WebBrowser
Instance Members
Properties inherited
AccessibilityObject:AccessibleObject

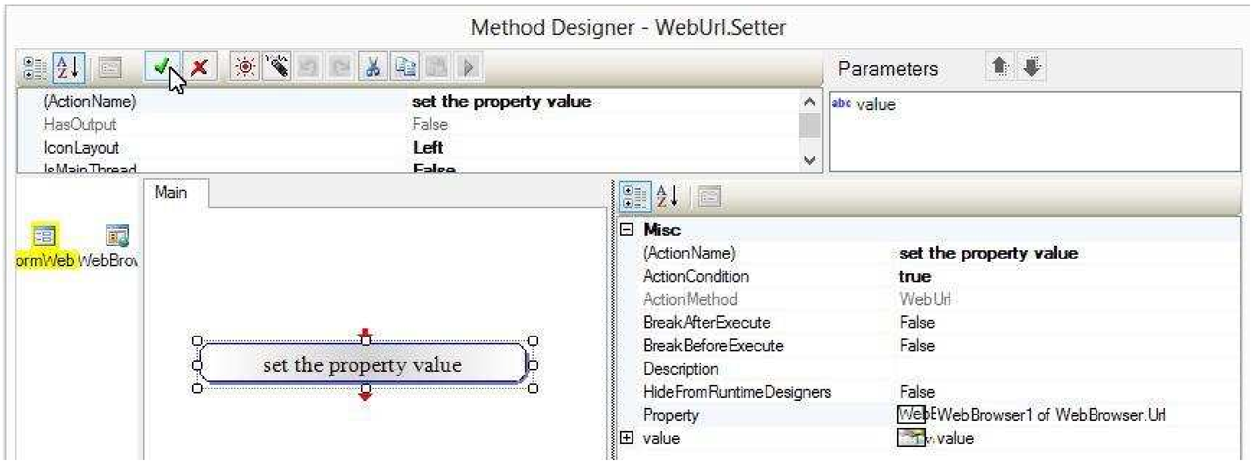
WebBrowser1 of WebBrowser.Url



Top:Int32
TopLevelControl:Control
Url:Uri
UseWaitCursor:Boolean
Version:Version

WebBrowser1 of WebBrowser.Url

Now, the new “value” for WebUrl property will not be passed to an internal value; instead, it will be passed to the Url property of the web browser control.



(ActionName) set the property value
HasOutput False
IconLayout Left
IsMainThread False
value

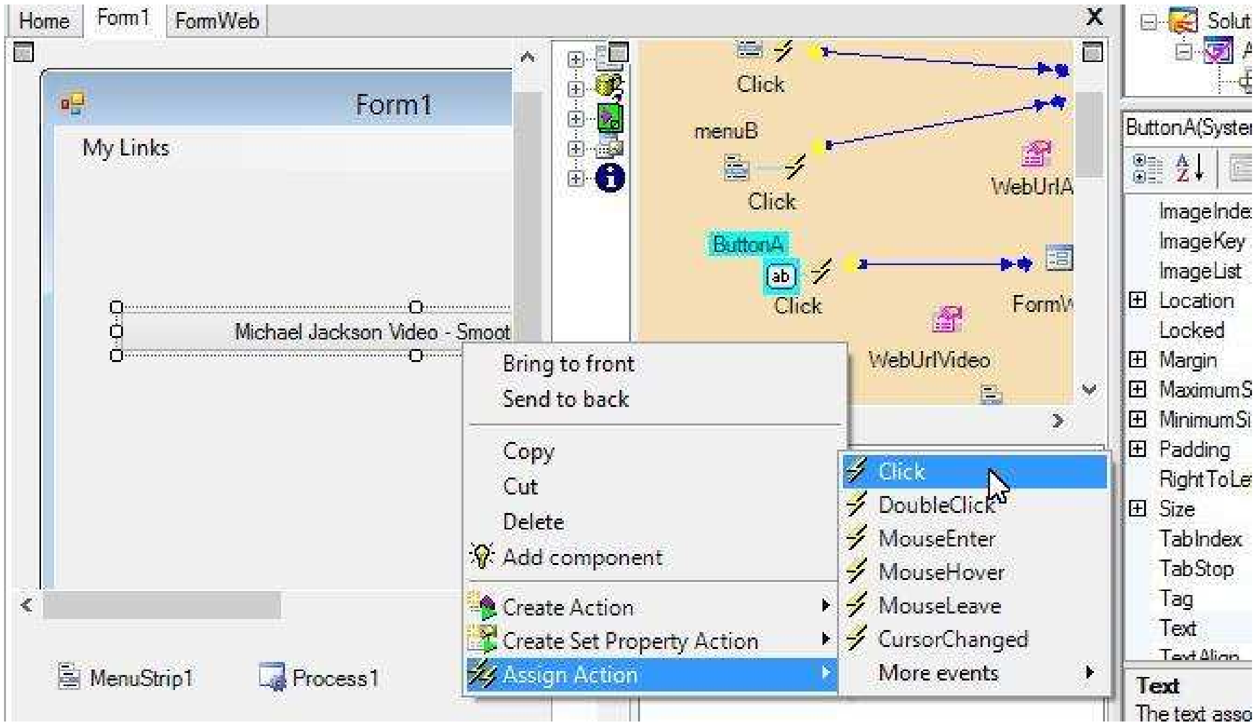
Misc
(ActionName) set the property value
ActionCondition true
ActionMethod WebUrl
BreakAfterExecute False
BreakBeforeExecute False
Description
HideFromRuntimeDesigners False
Property WebBrowser1 of WebBrowser.Url
value value

After modifying the Getter and Setter, the new property WebUrl behaves exactly as the Url property of the web browser control: when reading from property WebUrl, the value of the Url property of the web browser control is read; when writing to property WebUrl, the value of the Url property of the web browser control is written. The only difference is that WebUrl property is accessible by other forms, but Url property of the web browser control is not.

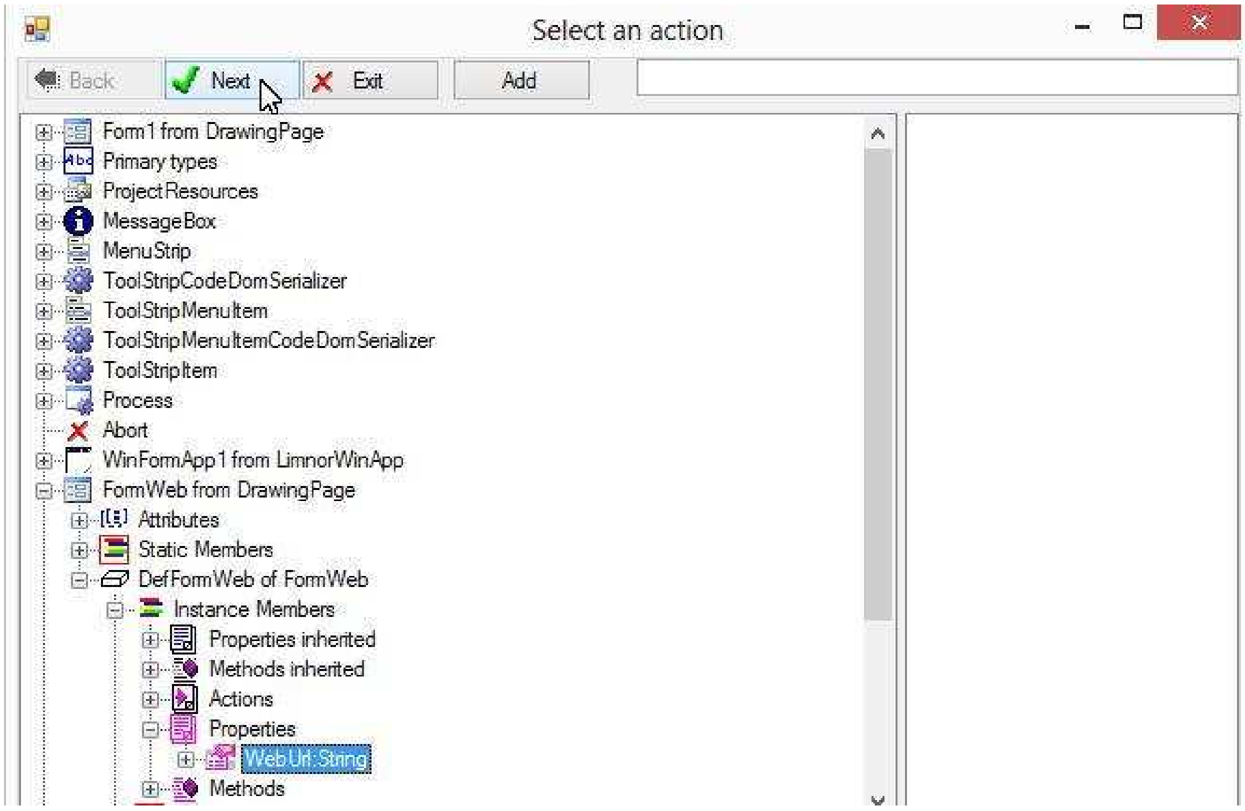
Now we may access the Url property from the main form when the button is clicked, via WebUrl property.

## Pass Software Value across forms

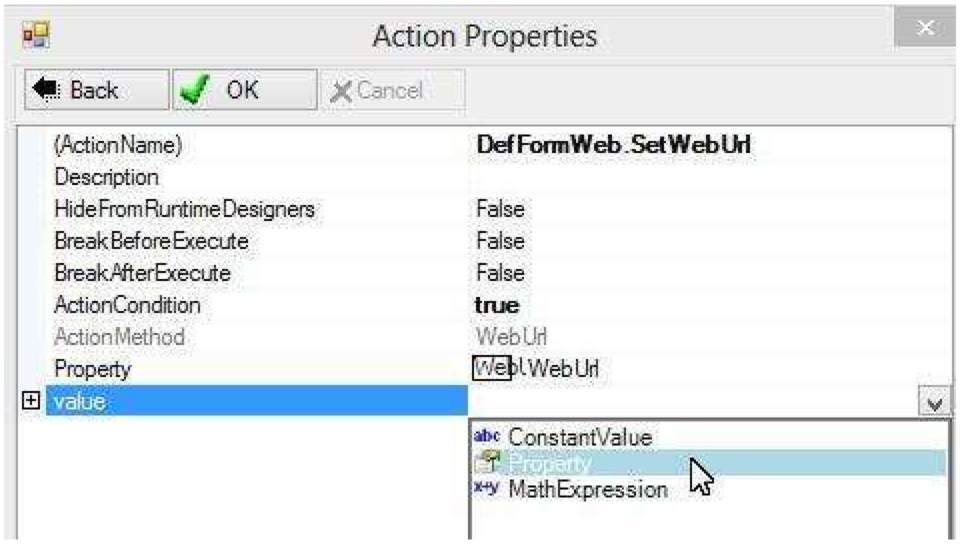
Clicking the button, our original programming is to show the web browser form. We also need to pass the user setting of the web URL to the web browser form. Right-click the button; choose “Assign Action”; choose “Click” event:



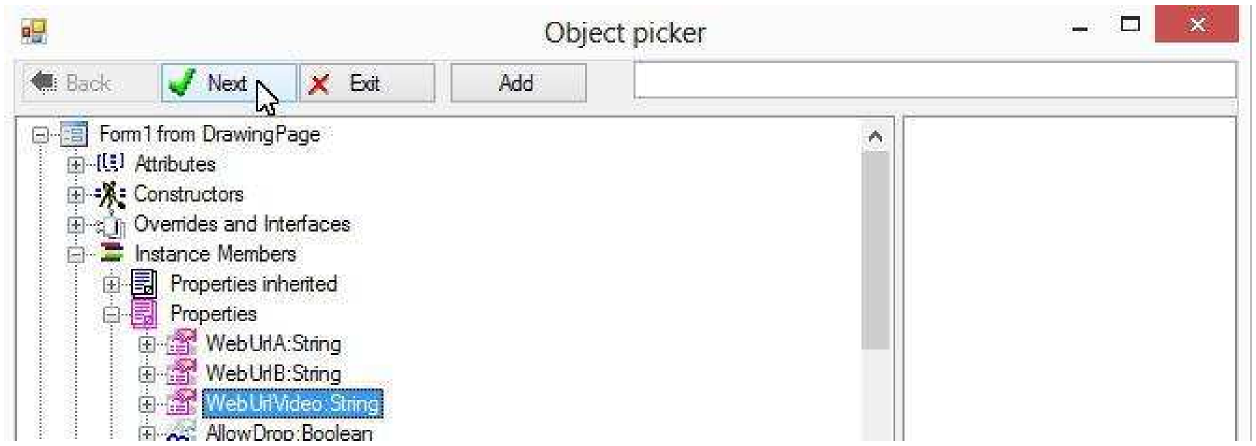
Choose the WebUrl property of the default instance of the web browser form:



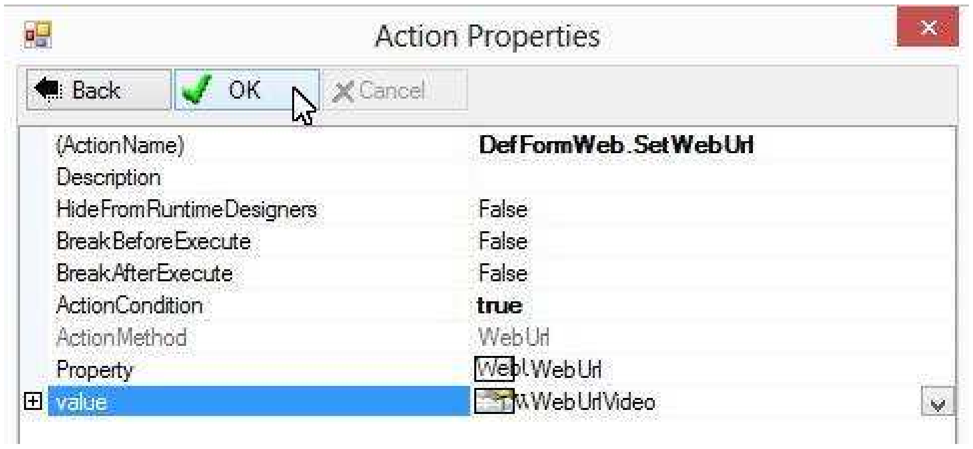
Select the property WebUrlVideo of the main form:



| | |
|---|---|
| (ActionName) | DefFormWeb.SetWebUrl |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | WebUrl |
| Property | WebUrl |
| value | |

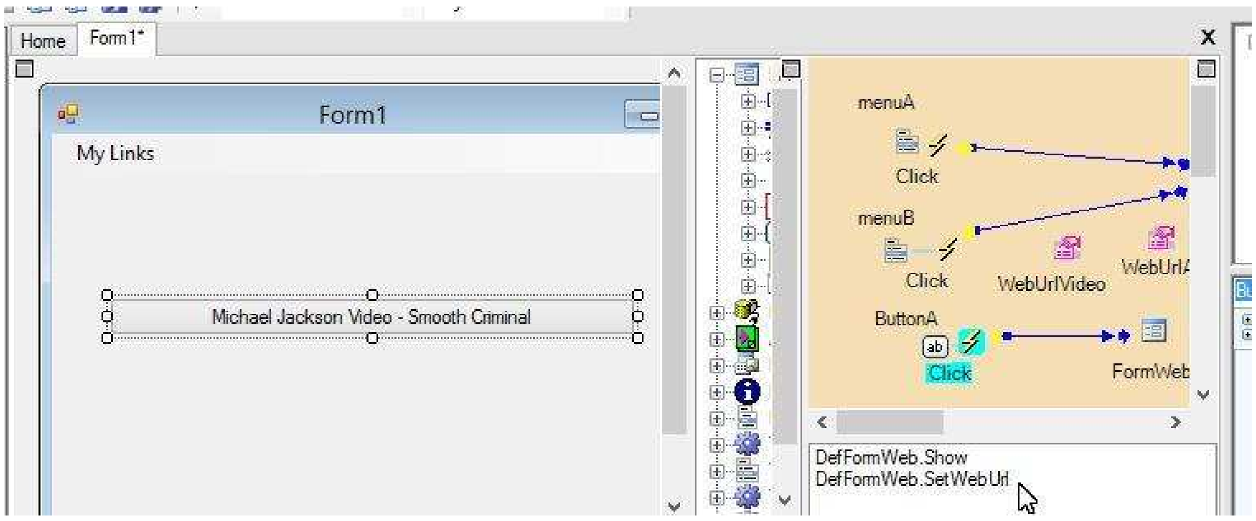


Click OK:



| | |
|---|---|
| (ActionName) | DefFormWeb.SetWebUrl |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | WebUrl |
| Property | WebUrl |
| value | WebUrlVideo |

The action is created and assigned to the button:



DefFormWeb.Show
DefFormWeb.SetWebUrl

## Summary of Software Values

Now we have replaced constants with writable properties to be set by users at runtime. Our Software Values are listed below.

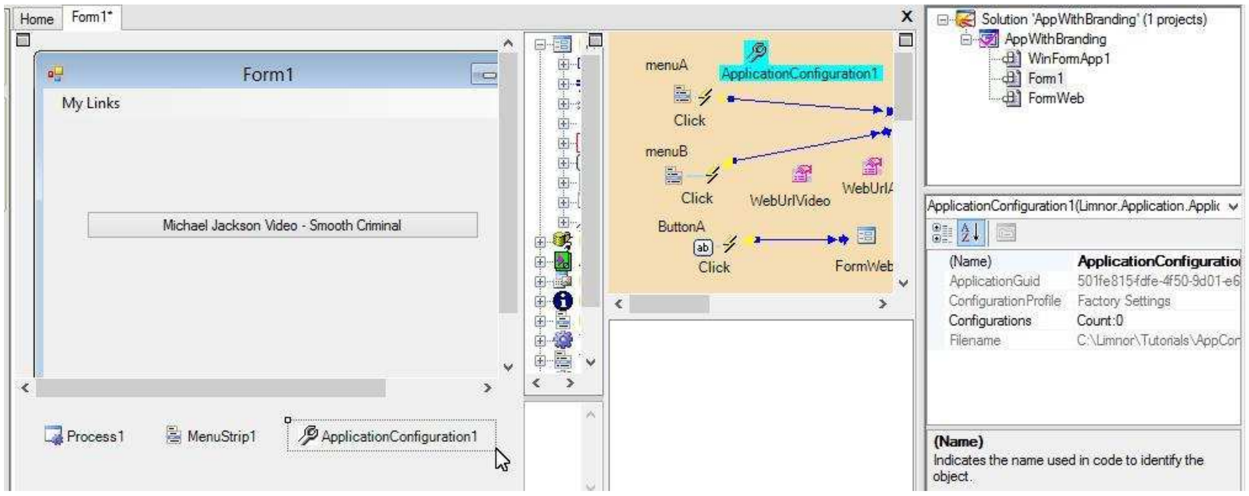
- Text property of menuA
- Property WebUrlA
- Text property of menuB
- Property WebUrlB
- Text property of buttonA
- Property WebUrlVideo

To allow branding of this software, we are going to add features to allow the users to set values for these properties. Each set of the values for the above properties forms one brand.

## Add Application Configuration Components

The Application Configuration Components are used to allow changing software properties at runtime.

For every class in your project, if it has properties you want them to be Software Values then you need to add an Application Configuration Component to the class. In our sample, we need to add an Application Configuration Component to our Form1.



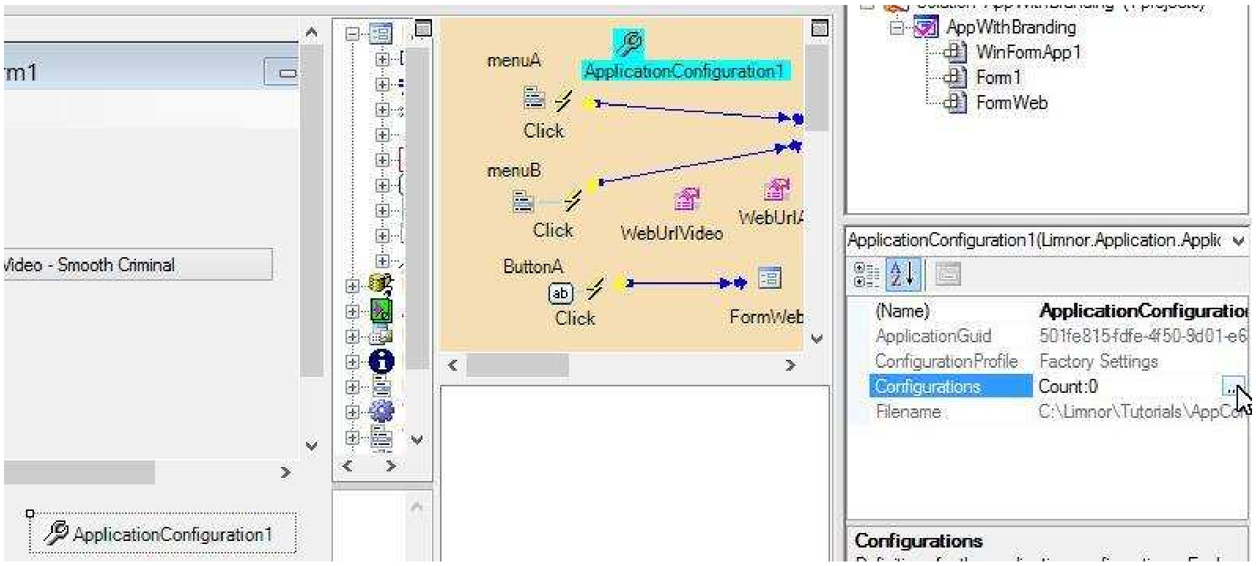
ApplicationConfiguration1(Limnor.Application.Applic

| | |
|---|---|
| (Name) | ApplicationConfiguratio |
| ApplicationGuid | 501fe815fdfe-4f50-9d01-e6 |
| ConfigurationProfile | Factory Settings |
| Configurations | Count:0 |
| Filename | C:\Limnor\Tutorials\AppCor |

**(Name)**
Indicates the name used in code to identify the object.

## Create Configuration Values

For each Software Value, we need to create a Configuration Value. To create configuration values, set Configurations property of an Application Configuration component:



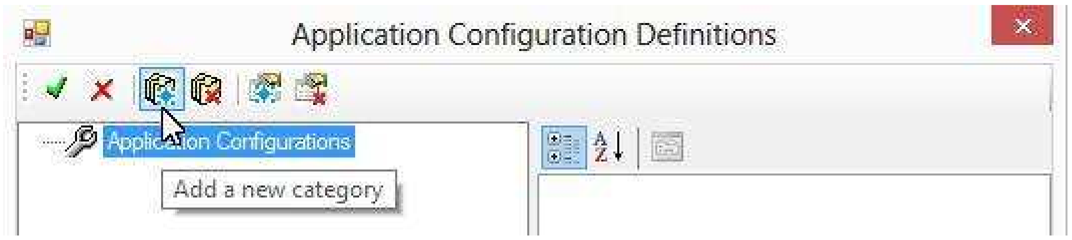
ApplicationConfiguration1(Limnor.Application.Applic

| | |
|---|---|
| (Name) | ApplicationConfiguratio |
| ApplicationGuid | 501fe815fdfe-4f50-9d01-e6 |
| ConfigurationProfile | Factory Settings |
| Configurations | Count:0 |
| Filename | C:\Limnor\Tutorials\AppCo |

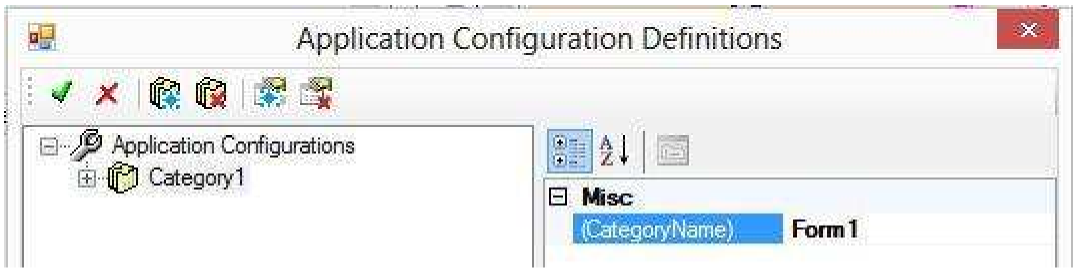
**Configurations**

## Create Categories

To help manage large amount of configuration values, you may create categories. Since we are going to create configuration values for Form1, let's create a category named Form1:


Application Configuration Definitions
Application Configurations
Add a new category

Rename the new category to Form1:

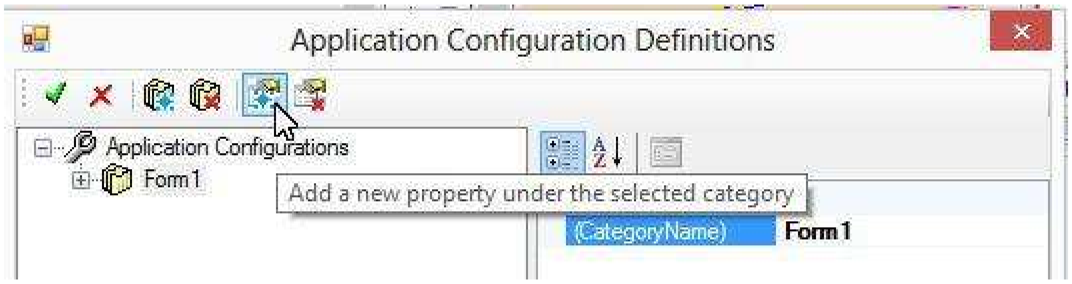

Application Configuration Definitions
Application Configurations
Category1
Misc

(CategoryName) Form1

Do not use following words for the names of the categories: AppKey, ApplicationGuid, Configuration, Configurations,ConfigurationProfile,Filename,ProfileName, ProfileType

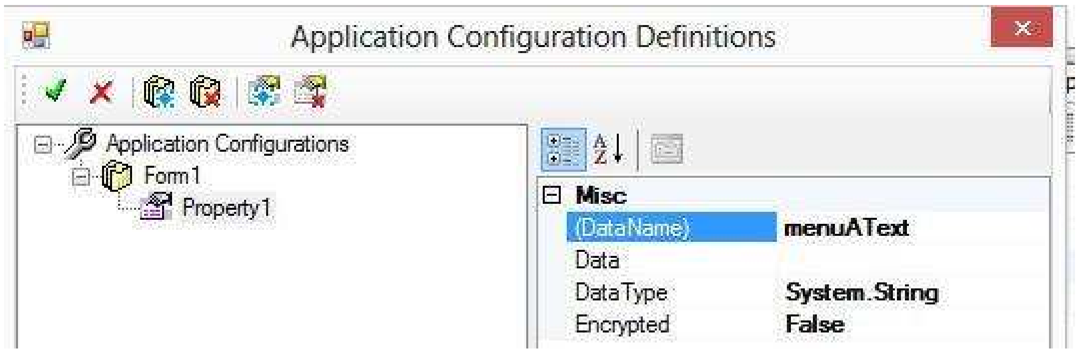
## Create Values

While a category is selected, we may create configuration values:


Application Configuration Definitions
Application Configurations
Form1
Add a new property under the selected category

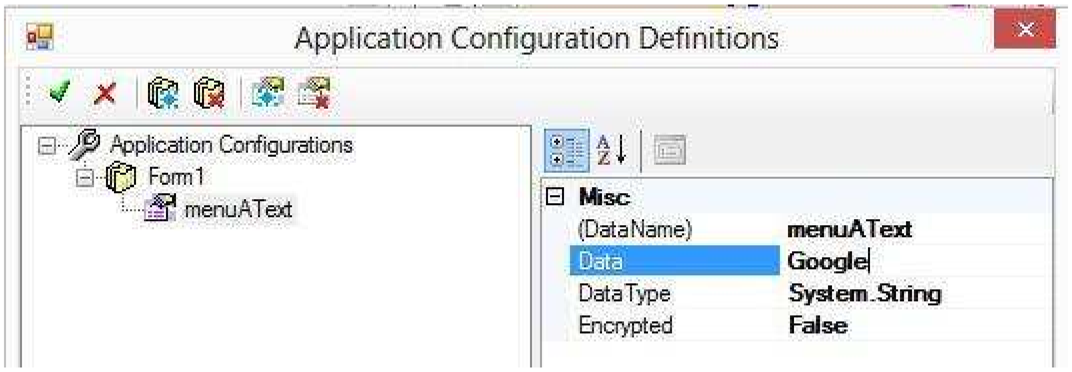
(CategoryName) Form1

Rename the new value to menuAText:


Application Configuration Definitions
Application Configurations
Form1
Property1
Misc

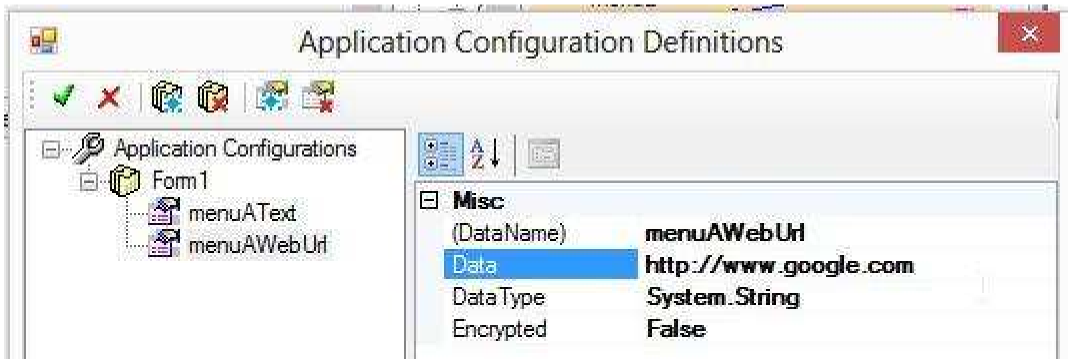
(DataName) menuAText
Data
DataType System.String
Encrypted False

Give it an initial value of Google:


Application Configuration Definitions
Application Configurations
Form1
menuAText
Misc

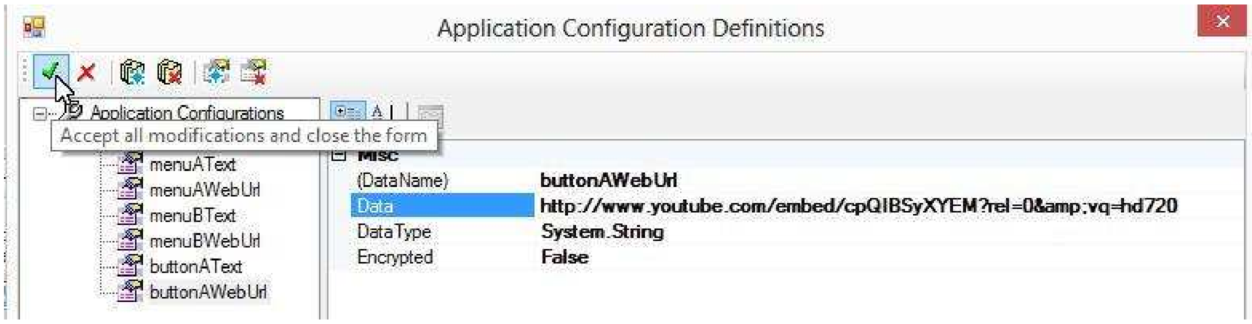
| | |
|---|---|
| (DataName) | menuAText |
| Data | Google |
| DataType | System.String |
| Encrypted | False |

Create another value named menuAWebUrl:


Application Configuration Definitions
Application Configurations
Form1
menuAText
menuAWebUrl
Misc

| | |
|---|---|
| (DataName) | menuAWebUrl |
| Data | http://www.google.com |
| DataType | System.String |
| Encrypted | False |

We need to create a configuration value for each Software Value:


Application Configuration Definitions
Application Configurations
Accept all modifications and close the form
menuAText
menuAWebUrl
menuBText
menuBWebUrl
buttonAText
buttonAWebUrl
Misc

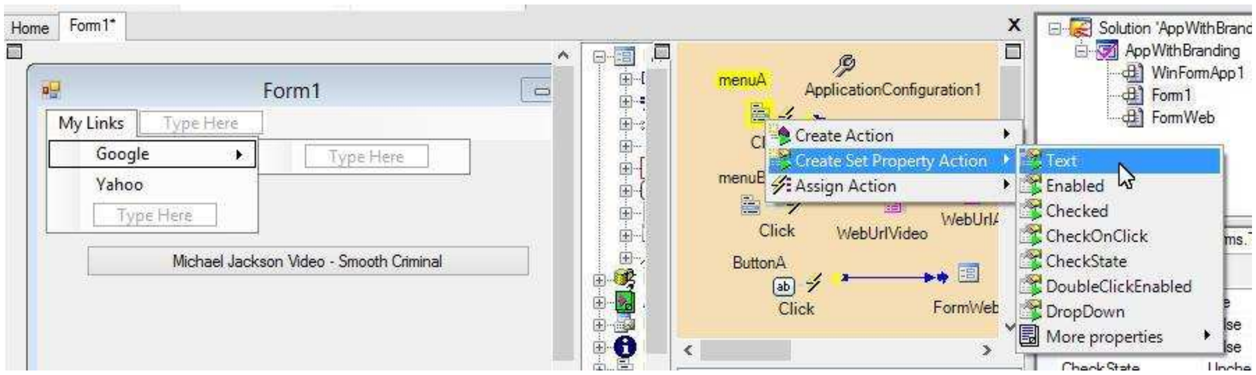
| | |
|---|---|
| (DataName) | buttonAWebUrl |
| Data | http://www.youtube.com/embed/cpQIBSyXYEM?rel=0&amp;vq=hd720 |
| DataType | System.String |
| Encrypted | False |

## Pass Configuration Values to Software Values

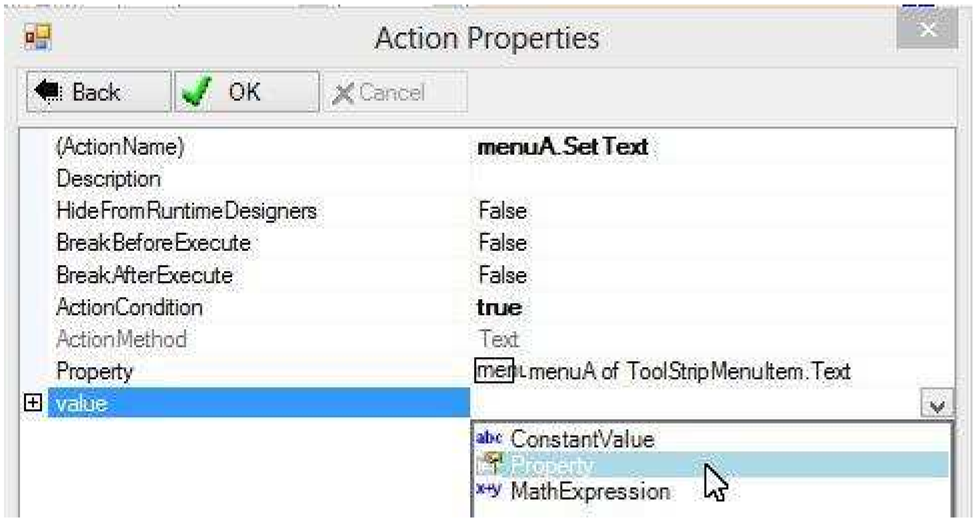
### Create data-passing actions

For each pair of Software Value and Configuration Value, we need to create an action to pass the Configuration Values to the Software Values.

Let’s create an action to pass menuAText to the Text property of menuA. Right-click menuA; choose “Create Set Property Action”; choose “Text” property (if you cannot find a property in the context menu then you may choose “More properties” to find it):

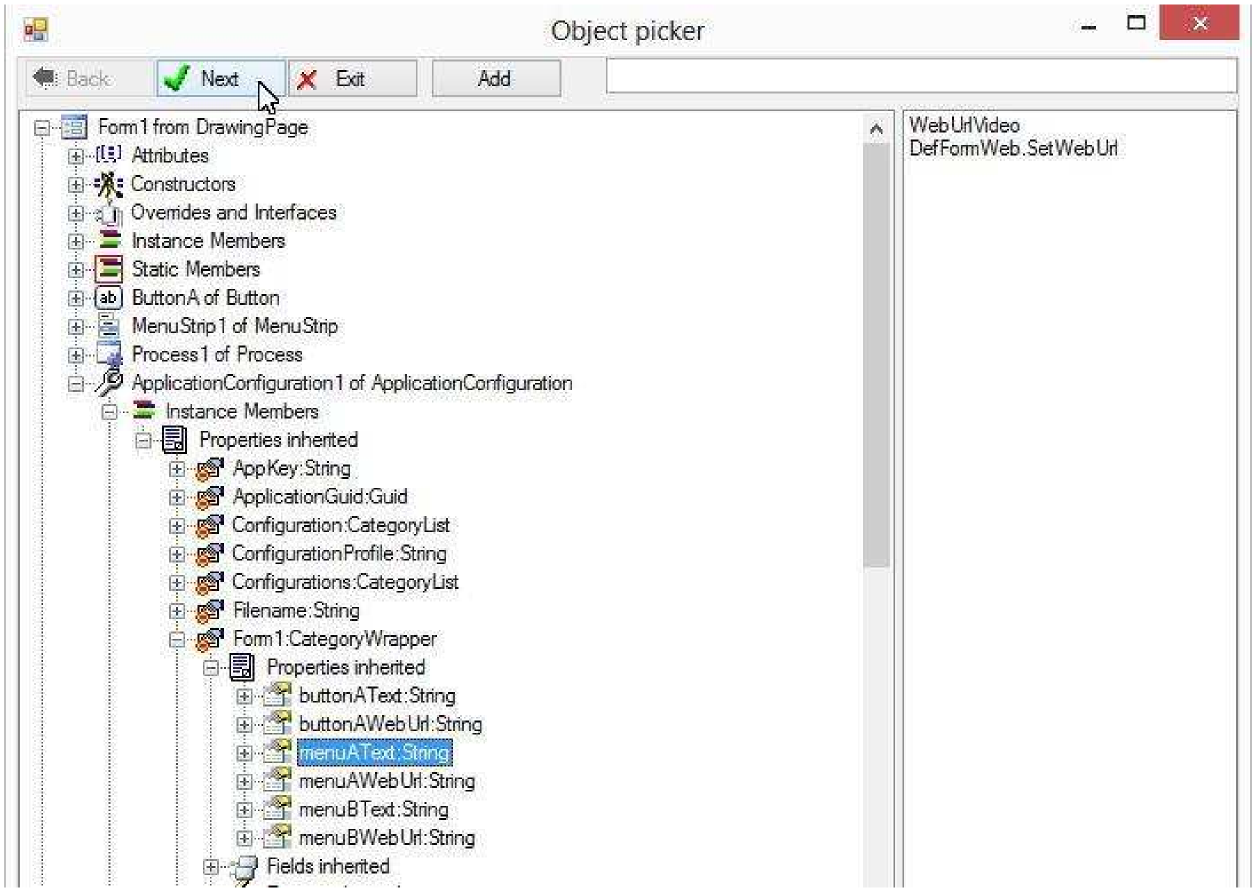


For “value” parameter of the action, choose the corresponding Configuration Value by selecting Property:



|  |  |
|---|---|
| (ActionName) | **menuA.SetText** |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | **true** |
| ActionMethod | Text |
| Property | menuA of ToolStripMenuItem.Text |
| value | |

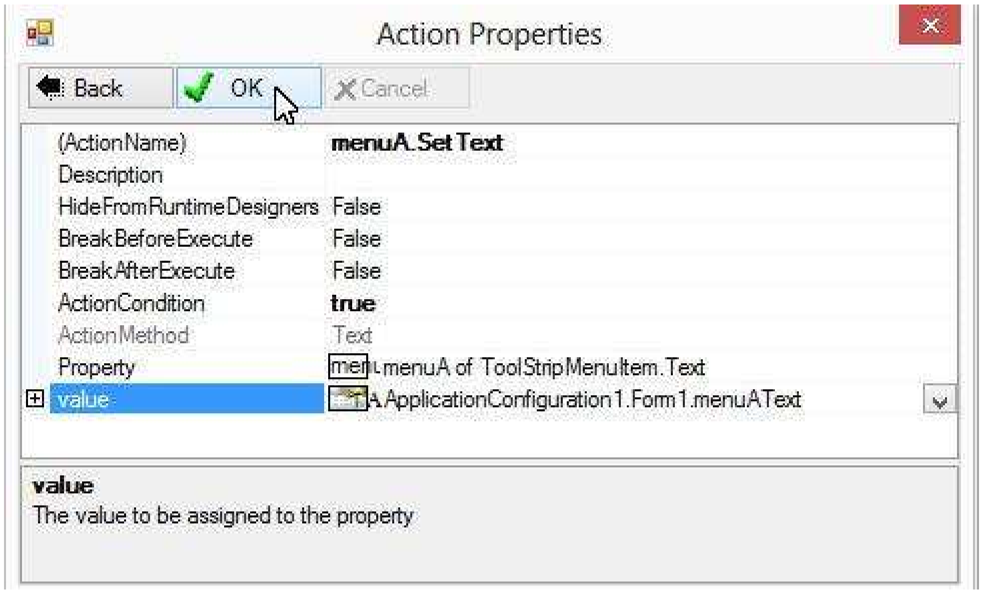
The corresponding Configuration Value is menuAText under category Form1:



Form1 from DrawingPage
- Attributes
- Constructors
- Overrides and Interfaces
- Instance Members
- Static Members
- ButtonA of Button
- MenuStrip1 of MenuStrip
- Process1 of Process
- ApplicationConfiguration1 of ApplicationConfiguration
  - Instance Members
    - Properties inherited
      - AppKey:String
      - ApplicationGuid:Guid
      - Configuration:CategoryList
      - ConfigurationProfile:String
      - Configurations:CategoryList
      - Filename:String
      - Form1:CategoryWrapper
        - Properties inherited
          - buttonAText:String
          - buttonAWebUrl:String
          - menuAText:String
          - menuAWebUrl:String
          - menuBText:String
          - menuBWebUrl:String
        - Fields inherited

WebUrlVideo
DefFormWeb.SetWebUrl

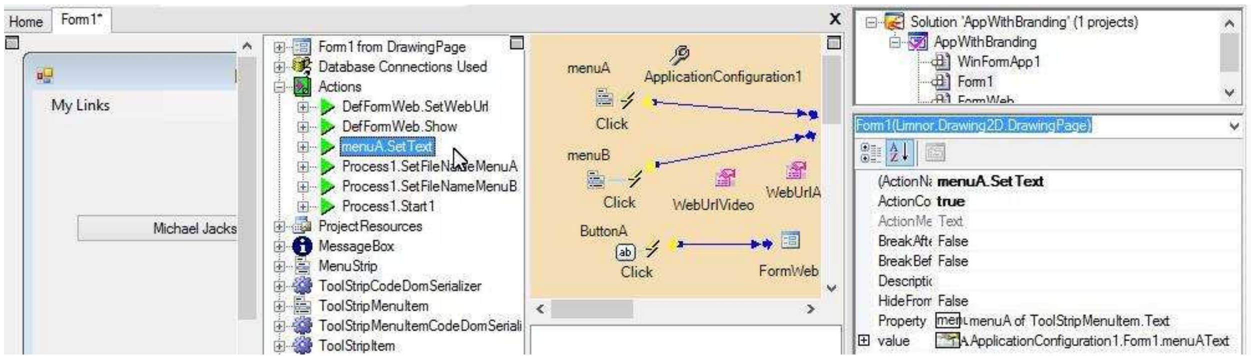
Click OK to finish creating this action:



| | |
|---|---|
| (ActionName) | menuA.Set Text |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | Text |
| Property | menuA of ToolStripMenuItem.Text |
| value | A.ApplicationConfiguration1.Form1.menuAText |

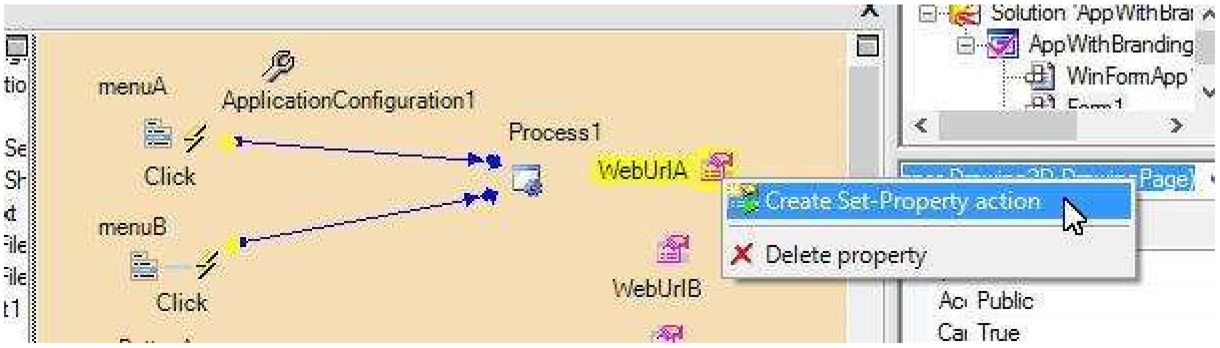
**value**
The value to be assigned to the property

The action is created. You can see it under Actions:

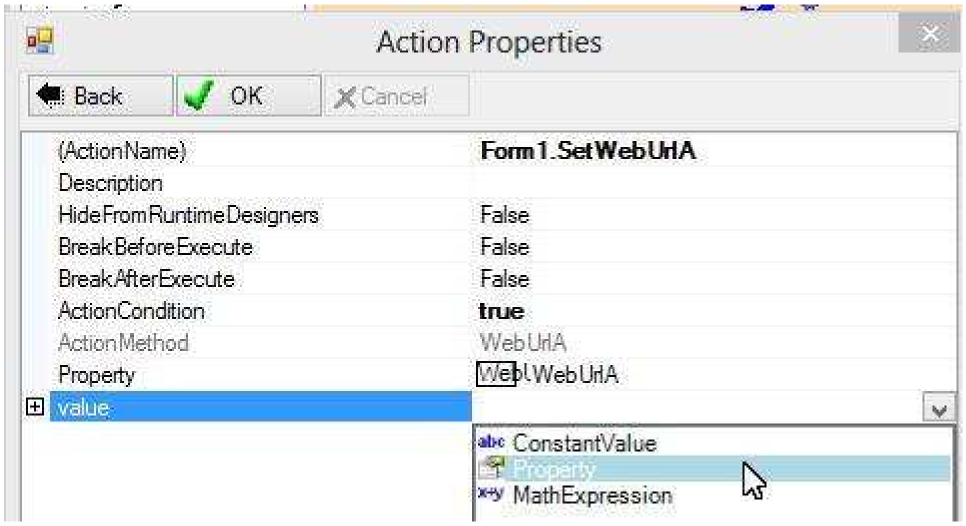


| | |
|---|---|
| (ActionN | menuA.SetText |
| ActionCo | true |
| ActionMe | Text |
| BreakAft | False |
| BreakBef | False |
| Descriptic | |
| HideFrom | False |
| Property | menuA of ToolStripMenuItem.Text |
| value | ApplicationConfiguration1.Form1.menuAText |

To create an action to pass a Configuration Value to a Property, right-click the property; choose "Create Set-Property action":

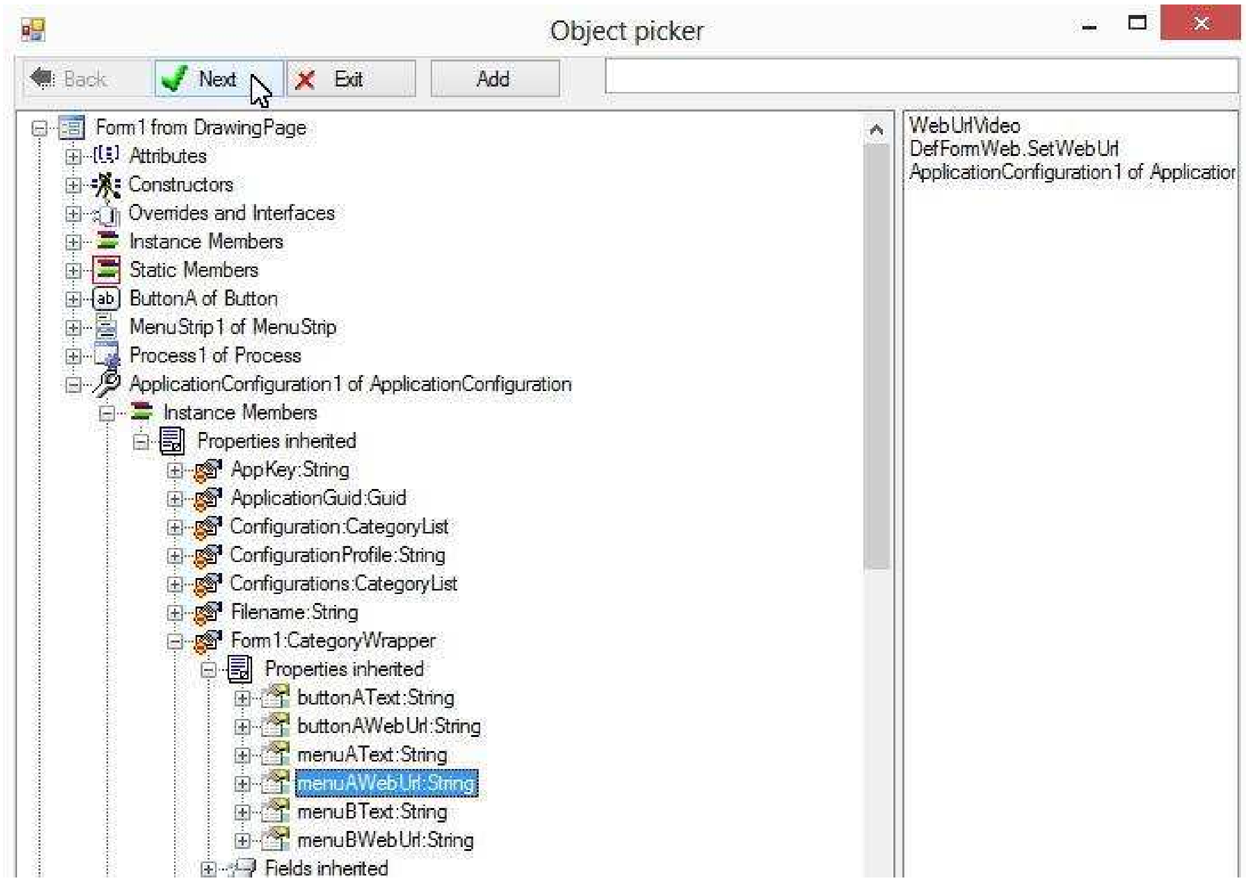


For "value" parameter of the action, choose the corresponding Configuration Value by selecting Property:



| | |
|---|---|
| (ActionName) | Form1.SetWebUrlA |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | WebUrlA |
| Property | Web WebUrlA |
| value | |

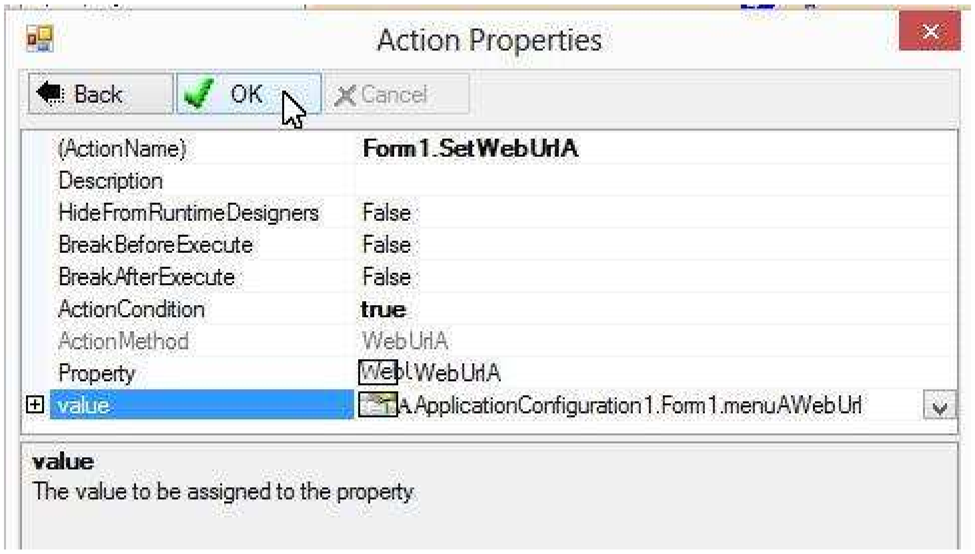
The corresponding Configuration Value is menuAWebUrl under category Form1:



```
Form1 from DrawingPage
  Attributes
  Constructors
  Overrides and Interfaces
  Instance Members
  Static Members
  ButtonA of Button
  MenuStrip1 of MenuStrip
  Process1 of Process
  ApplicationConfiguration1 of ApplicationConfiguration
    Instance Members
      Properties inherited
        AppKey:String
        ApplicationGuid:Guid
        Configuration:CategoryList
        ConfigurationProfile:String
        Configurations:CategoryList
        Filename:String
        Form1:CategoryWrapper
          Properties inherited
            buttonAText:String
            buttonAWebUrl:String
            menuAText:String
            menuAWebUrl:String
            menuBText:String
            menuBWebUrl:String
          Fields inherited
```

```
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicatior
```

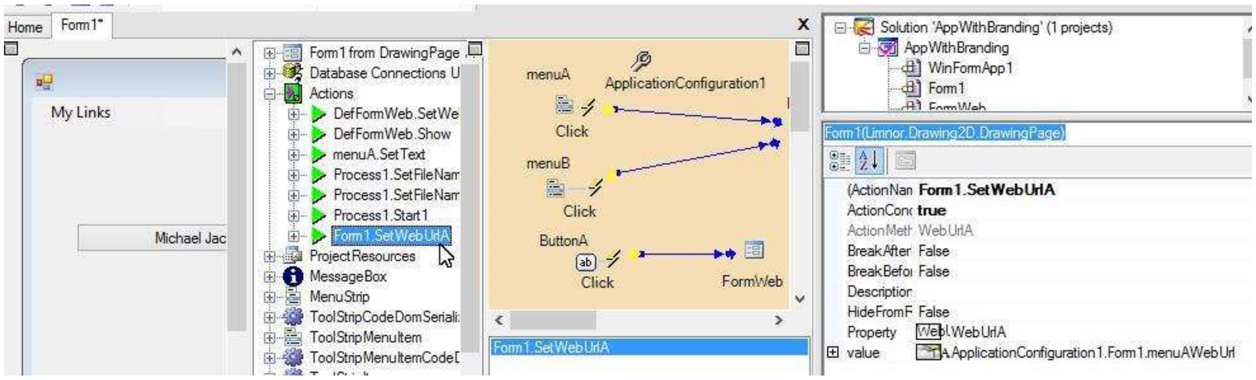
Click OK to finish creating this action:



| | |
|---|---|
| (ActionName) | **Form1.SetWebUrlA** |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | **true** |
| ActionMethod | WebUrlA |
| Property | WebUrl.WebUrlA |
| value | A.ApplicationConfiguration1.Form1.menuAWebUrl |

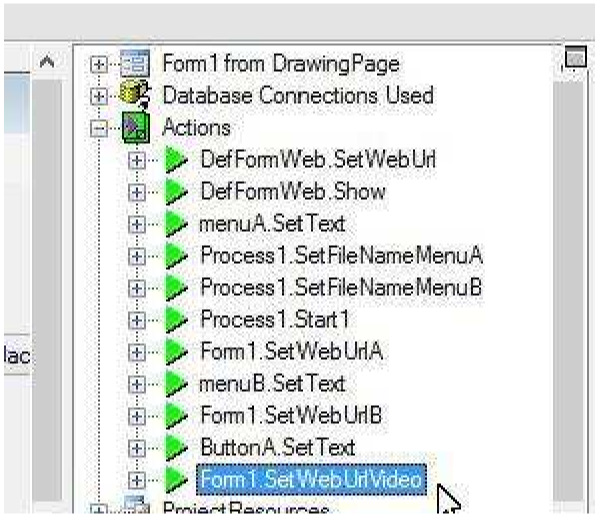
**value**
The value to be assigned to the property

The action is created. You can see it under Actions:



| | |
|---|---|
| (ActionNan | **Form1.SetWebUrlA** |
| ActionCon | **true** |
| ActionMeth | WebUrlA |
| BreakAfter | False |
| BreakBefo | False |
| Description | |
| HideFromF | False |
| Property | WebUrlA |
| value | A ApplicationConfiguration1.Form1.menuAWebUrl |

We need to create one action for each pair of Software Value and Configuration Value to pass the Configuration Value to the Software Value, as we did above. We are not go into details.

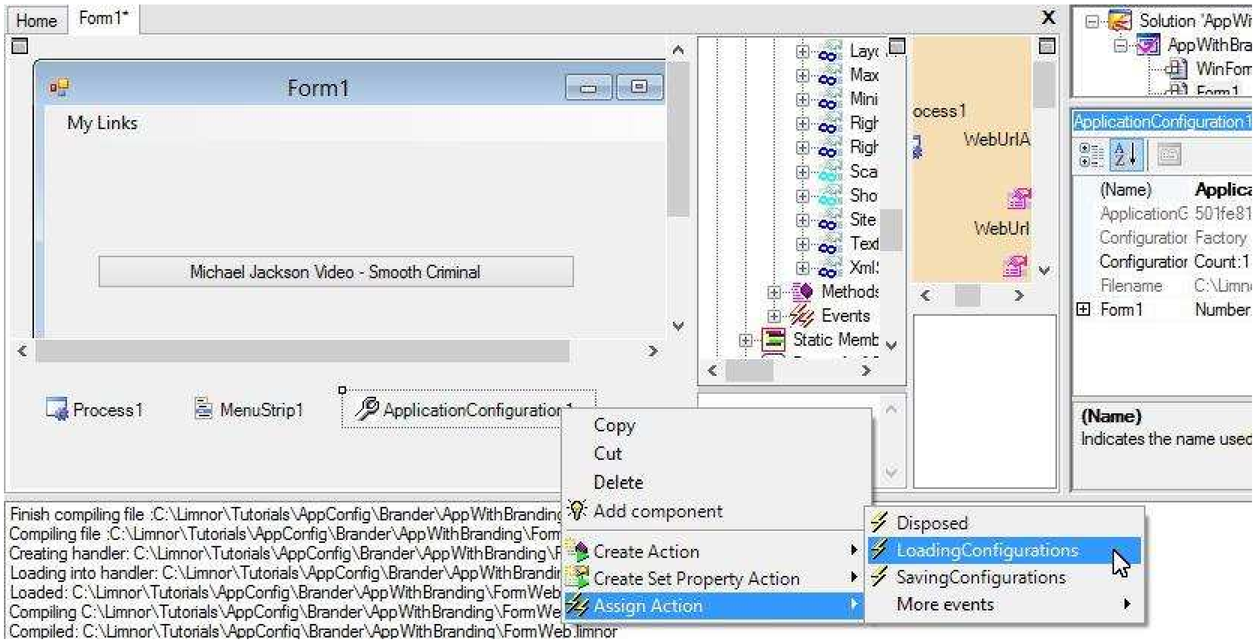

Form1 from DrawingPage
Database Connections Used
Actions
DefFormWeb.SetWebUrl
DefFormWeb.Show
menuA.SetText
Process1.SetFileNameMenuA
Process1.SetFileNameMenuB
Process1.Start1
Form1.SetWebUrlA
menuB.SetText
Form1.SetWebUrlB
ButtonA.SetText
Form1.SetWebUrlVideo
ProjectResources

In summary, we have done following design and programming.

| Software Value | Configuration Value | Action |
|---|---|---|
| Text of menuA | menuAText | menuA.SetText |
| WebUrlA | menuAWebUrl | Form1.SetWebUrlA |
| Text of menuB | menuBText | menuB.SetText |
| WebUrlB | menuBWebUrl | Form1.SetWebUrlB |
| Text of buttonA | buttonAText | buttonA.SetText |
| WebUrlVideo | buttonAWebUrl | Form1.SetWebUrlVideo |

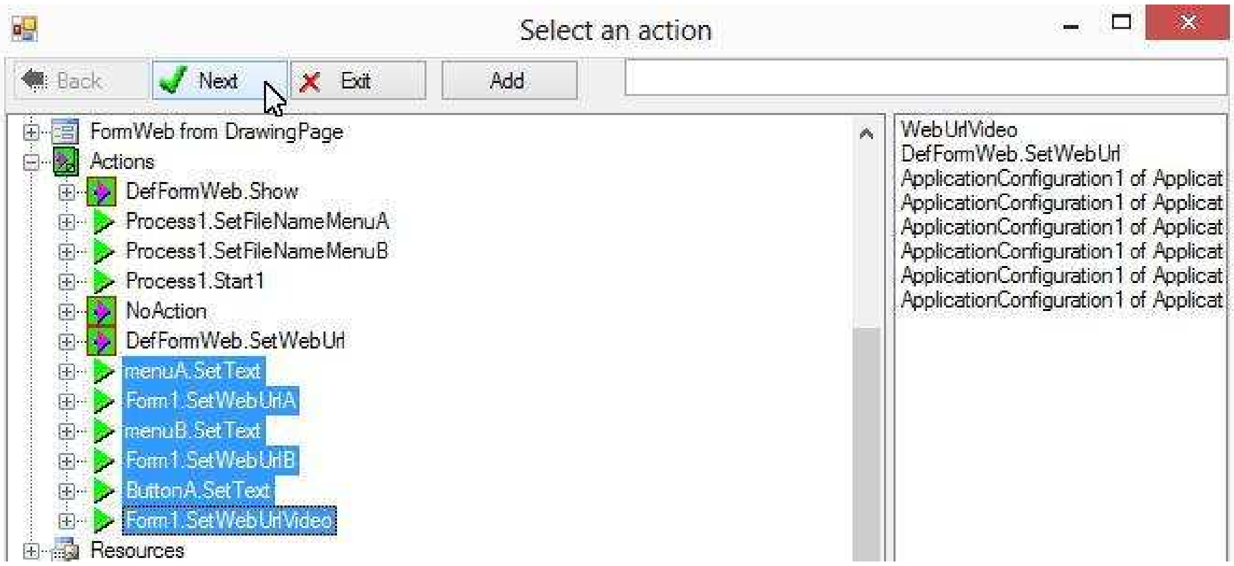
## Use LoadingConfigurations event

We have created data-passing actions. We want these actions to be executed at the event of LoadingConfigurations. Right-click the Application Configuration component; choose “Assign Action”; choose “LoadingConfigurations” event:

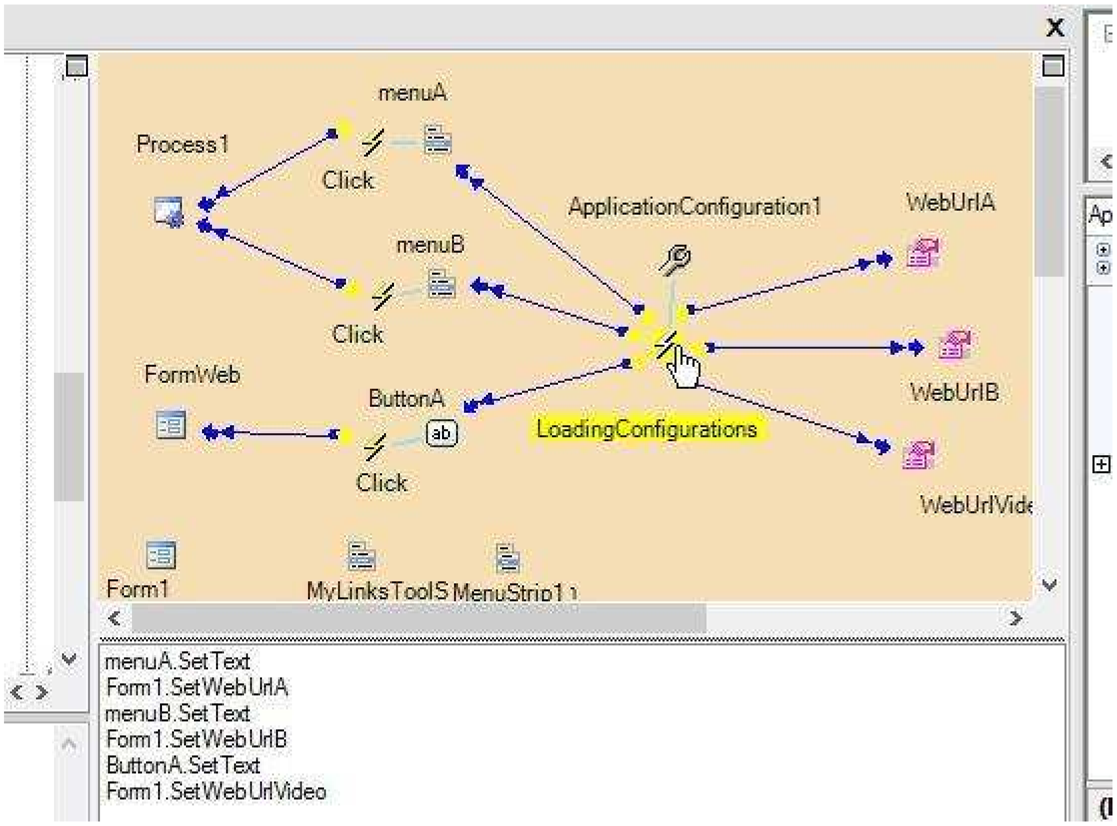


```
Finish compiling file :C:\Limnor\Tutorials\AppConfig\Brander\AppWithBranding
Compiling file :C:\Limnor\Tutorials\AppConfig\Brander\AppWithBranding\Form
Creating handler: C:\Limnor\Tutorials\AppConfig\Brander\AppWithBranding\F
Loading into handler: C:\Limnor\Tutorials\AppConfig\Brander\AppWithBrandir
Loaded: C:\Limnor\Tutorials\AppConfig\Brander\AppWithBranding\FormWeb
Compiling C:\Limnor\Tutorials\AppConfig\Brander\AppWithBranding\FormWe
Compiled: C:\Limnor\Tutorials\AppConfig\Brander\AppWithBranding\FormWeb.limnor
```

Choose the 6 data-passing actions:


Select an action
Back
Next
Exit
Add
FormWeb from DrawingPage
Actions
DefFormWeb.Show
Process1.SetFileNameMenuA
Process1.SetFileNameMenuB
Process1.Start1
NoAction
DefFormWeb.SetWebUrl
menuA.SetText
Form1.SetWebUrlA
menuB.SetText
Form1.SetWebUrlB
ButtonA.SetText
Form1.SetWebUrlVideo
Resources
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat

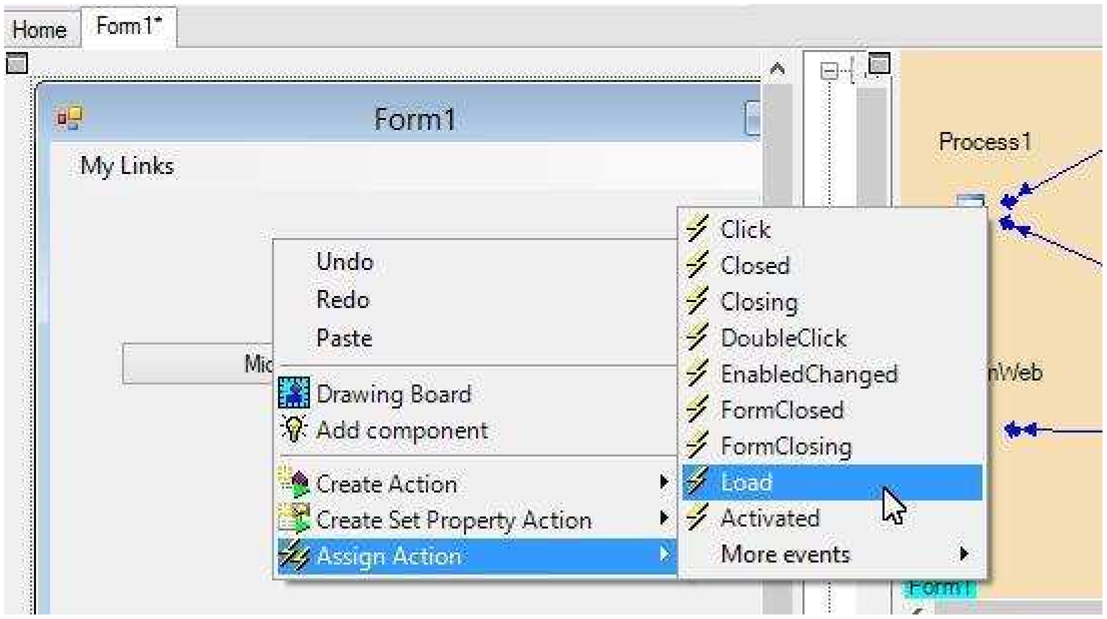
These actions are assigned to event LoadingConfigurations:



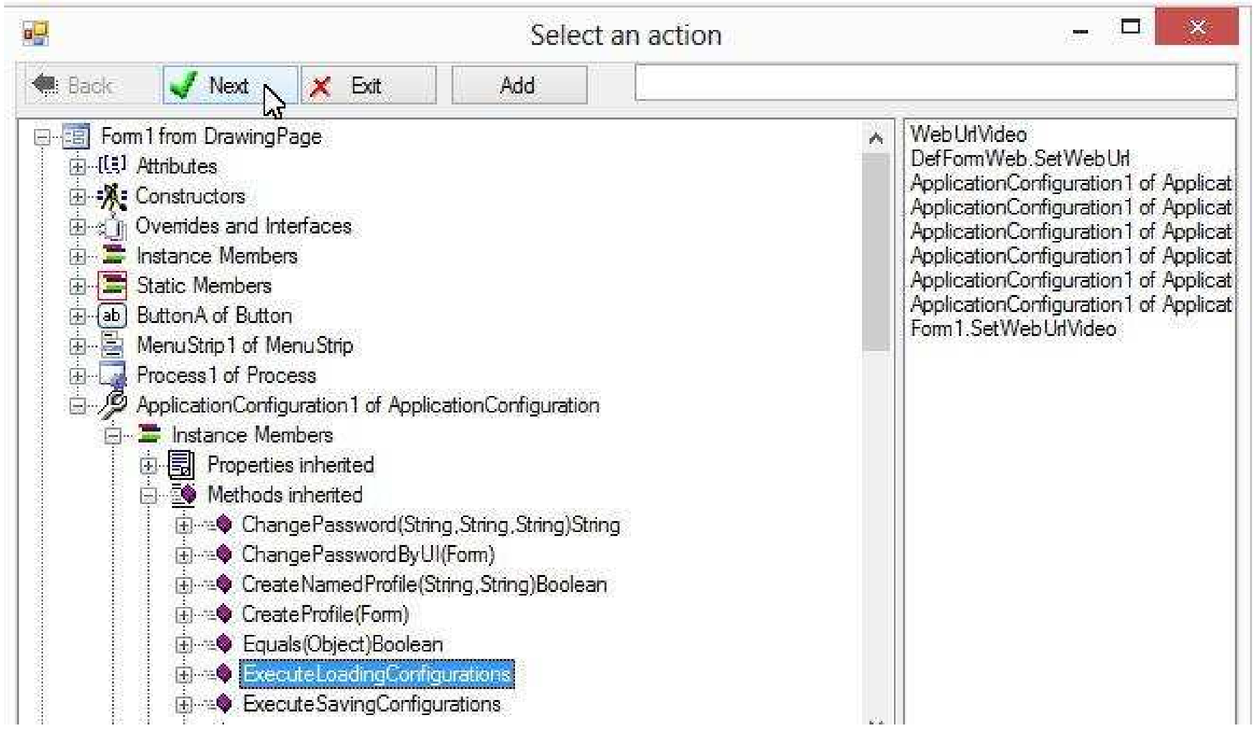
menuA.SetText
Form1.SetWebUrlA
menuB.SetText
Form1.SetWebUrlB
ButtonA.SetText
Form1.SetWebUrlVideo

## Apply Configurations

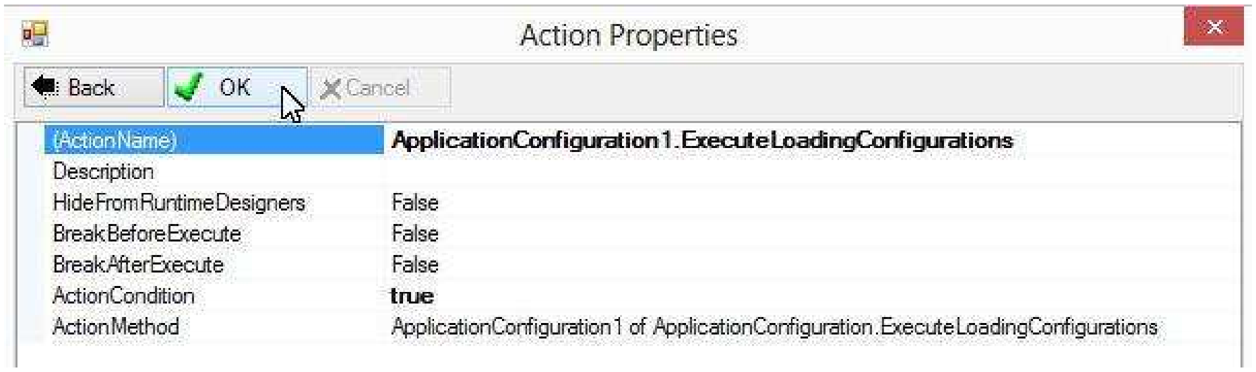
We want to apply the configurations when Fom1 is loaded. So, we may do it at the Load event of the Form1. Right-click Form1; choose “Assign Action”; choose “Load” event:



Choose ExecuteLoadingConfigurations method of the Application Configuration component:

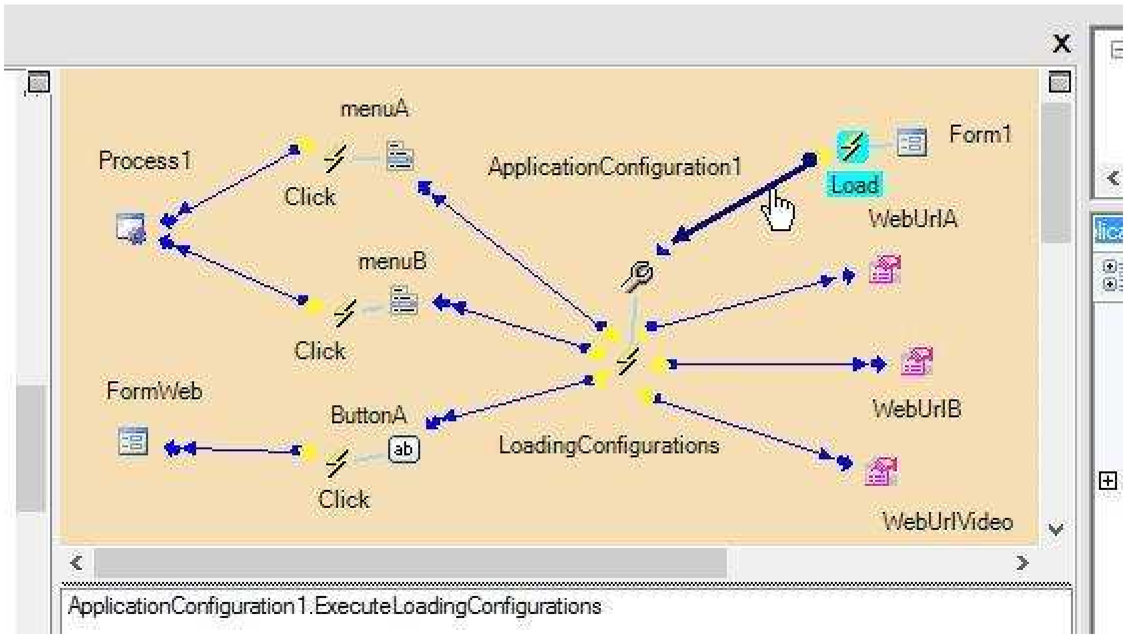

Select an action
Back
Next
Exit
Add
Form1 from DrawingPage
Attributes
Constructors
Overrides and Interfaces
Instance Members
Static Members
ButtonA of Button
MenuStrip1 of MenuStrip
Process1 of Process
ApplicationConfiguration1 of ApplicationConfiguration
Instance Members
Properties inherited
Methods inherited
ChangePassword(String,String,String)String
ChangePasswordByUI(Form)
CreateNamedProfile(String,String)Boolean
CreateProfile(Form)
Equals(Object)Boolean
ExecuteLoadingConfigurations
ExecuteSavingConfigurations
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
Form1.SetWebUrlVideo

Click OK:



| | |
|---|---|
| (ActionName) | ApplicationConfiguration1.ExecuteLoadingConfigurations |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfiguration.ExecuteLoadingConfigurations |

An action of ExecuteLoadingConfigurations is created and assigned to the Load event of Form1:



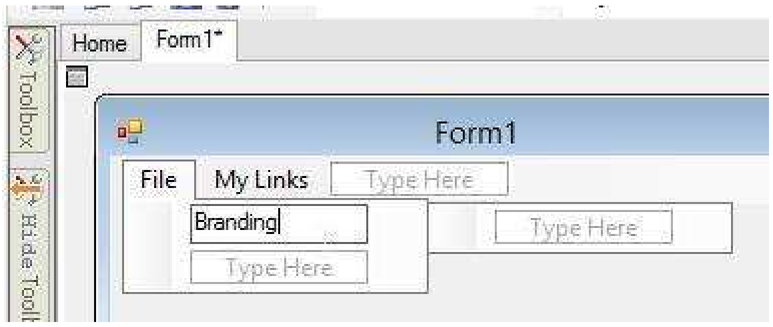
By the above programming, the program runs in the following event sequence: Form1 is loaded and generates event "Load"; "Load" event triggers action "ExecuteLoadingConfigurations" which fires event LoadingConfigurations of the Application Configuration component; event "LoadingConfigurations" triggers 6 data-passing actions which pass Configuration Values to Software Values. Thus software properties are modified at runtime.

## Enable Users to Set Configuration Values

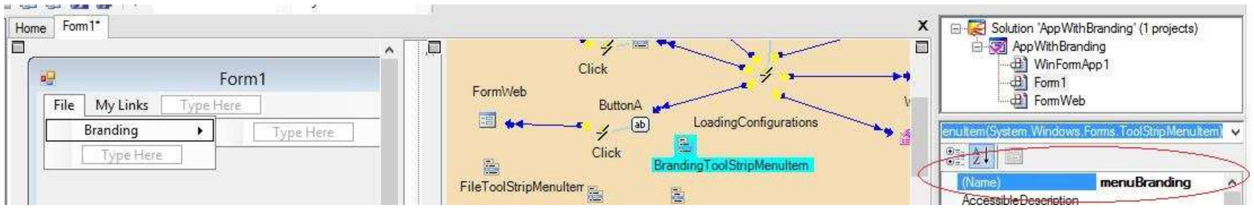
Our previous programming has enabled applying configuration values at runtime. Now we need to enable the users to set the configuration values.

### SetValues action

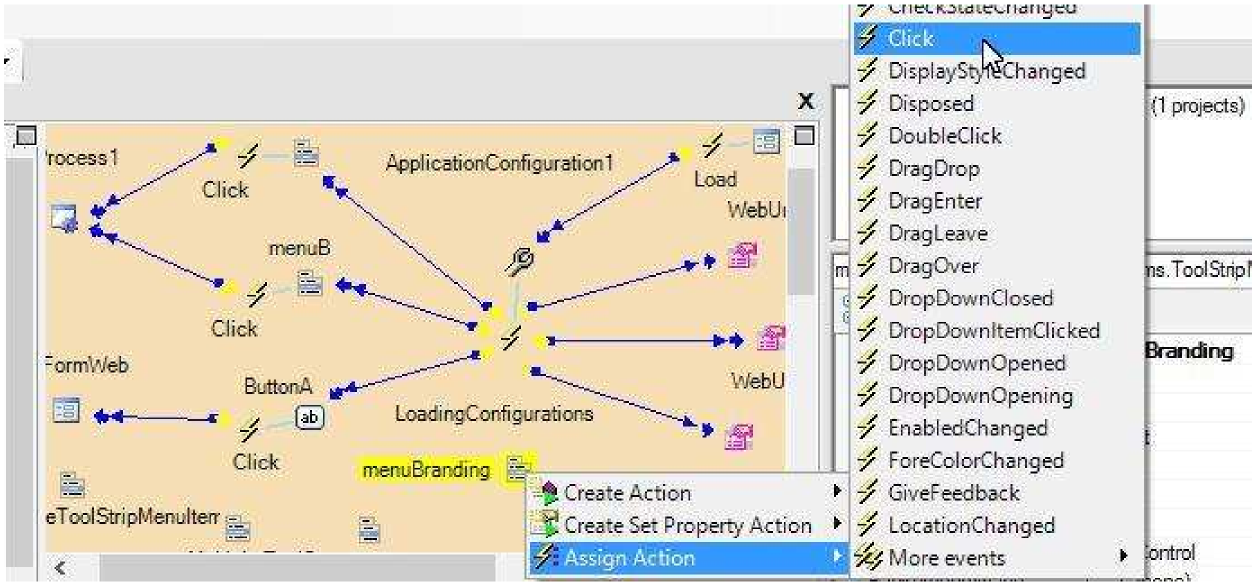
SetValues method of an Application Configuration component enables the users to set configuration values. In this sample, let's use a menu item to let the user execute this method.



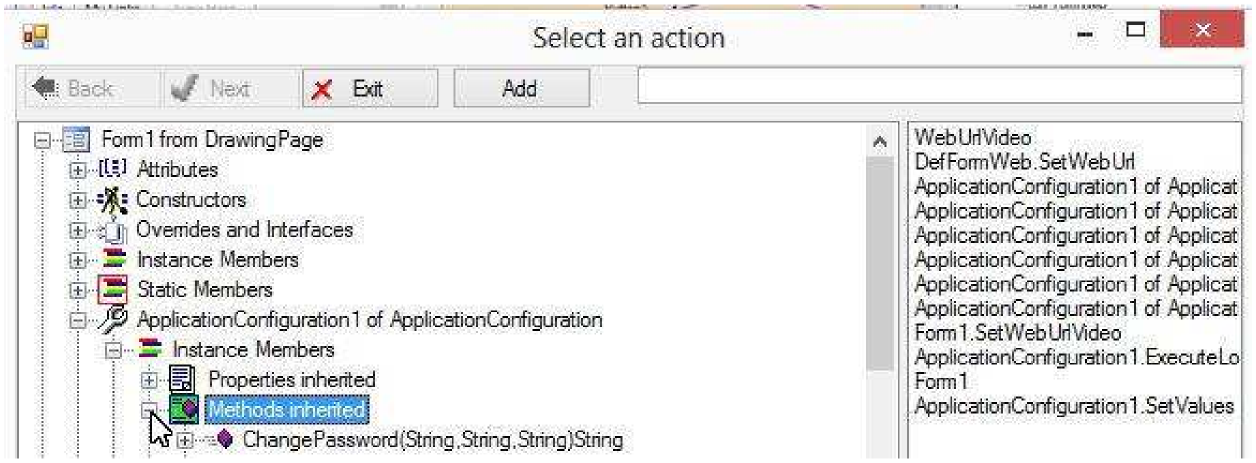
Rename the new menu item to menuBranding:

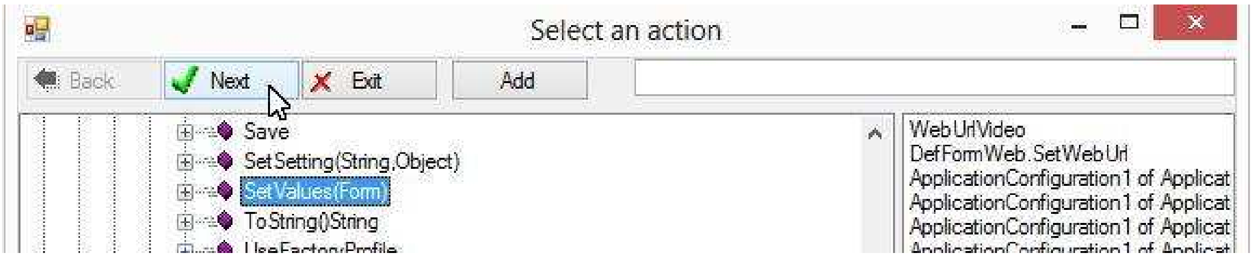


Right-click the new menu; choose “Assign Action”; choose “Click” event:

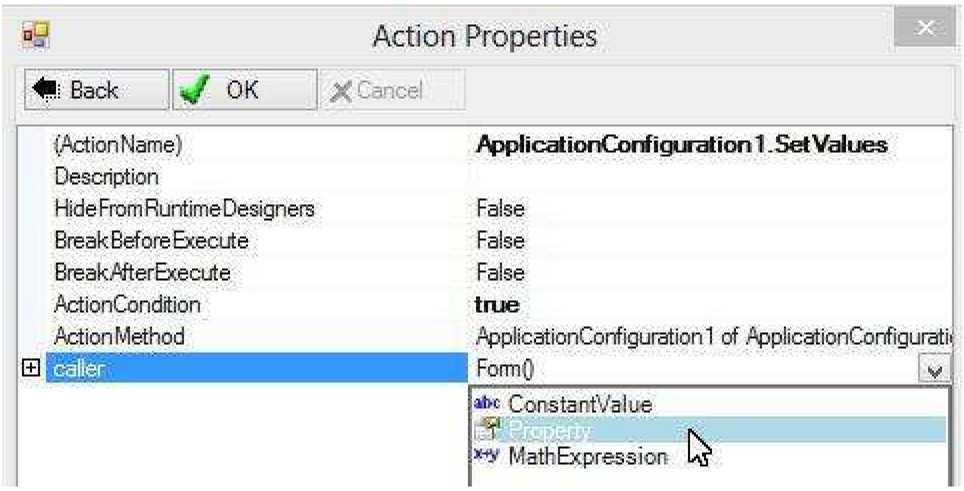


Choose SetValues method of the Application Configuration component:

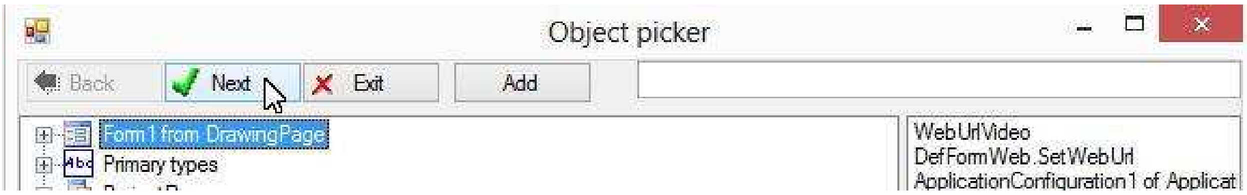

Select an action
Back
Next
Exit
Add
Form1 from DrawingPage
Attributes
Constructors
Overrides and Interfaces
Instance Members
Static Members
ApplicationConfiguration1 of ApplicationConfiguration
Instance Members
Properties inherited
Methods inherited
ChangePassword(String,String,String)String
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
Form1.SetWebUrlVideo
ApplicationConfiguration1.ExecuteLo
Form1
ApplicationConfiguration1.SetValues


Select an action
Back
Next
Exit
Add
Save
SetSetting(String,Object)
SetValues(Form)
ToString()String
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat

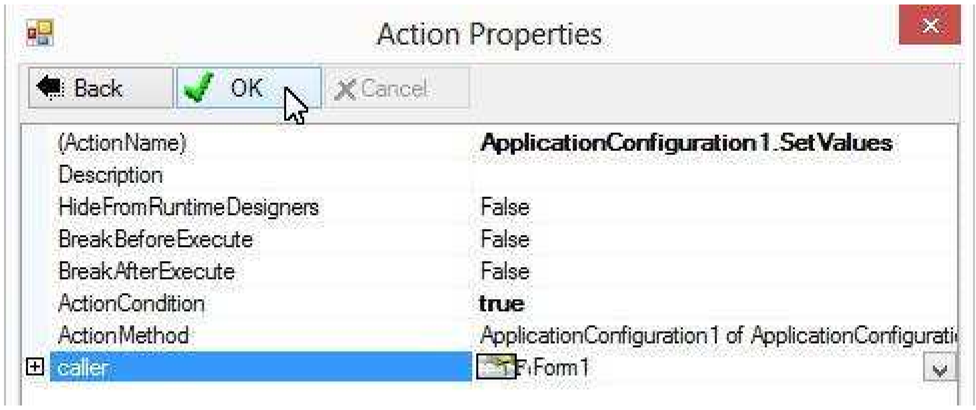
This action will launch a dialogue box. The "caller" parameter of the action indicates the owner of the dialogue box. Set "caller" to Form1:



| | |
|---|---|
| (ActionName) | ApplicationConfiguration1.SetValues |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfigurati |
| caller | Form() |

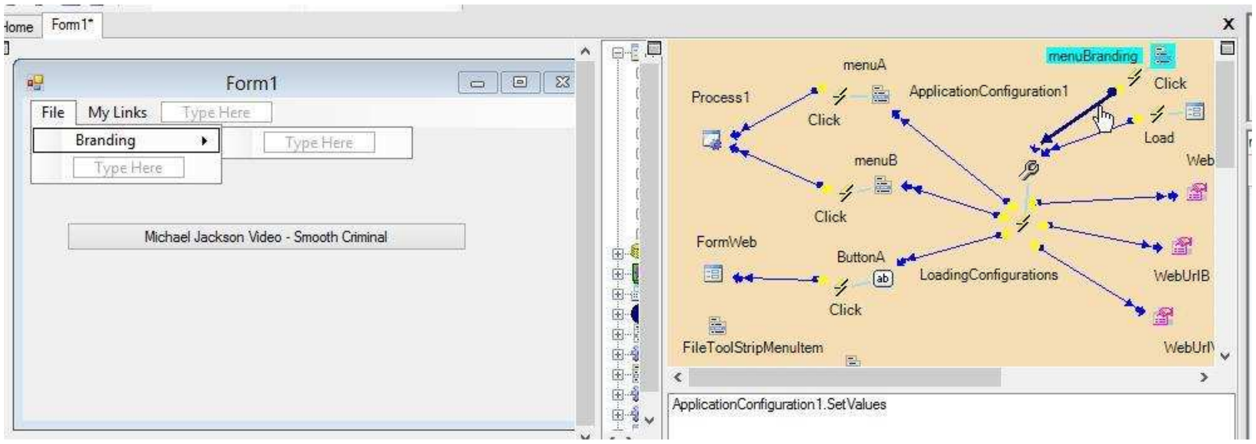


Click OK:



| | |
|---|---|
| (ActionName) | ApplicationConfiguration1.SetValues |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfigurati |
| caller | F:Form1 |

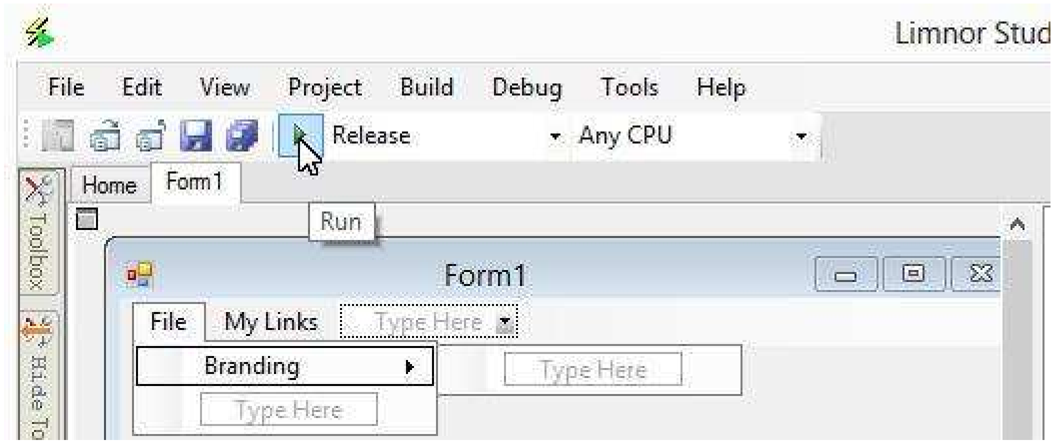
The action is created and assigned to the menu:



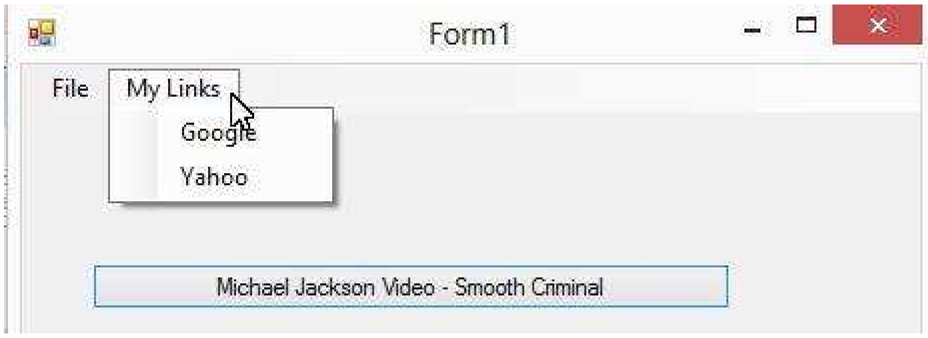
## User Test

### Verify original functionality

Now we basically finished weaving branding into our software. We may do testing now.



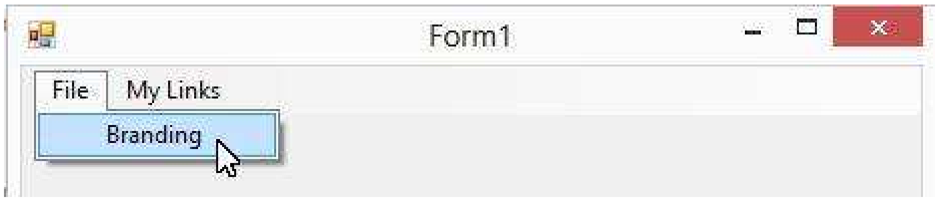
Form1 appears. We can see that all components appear as the original programming without weaving branding into it.



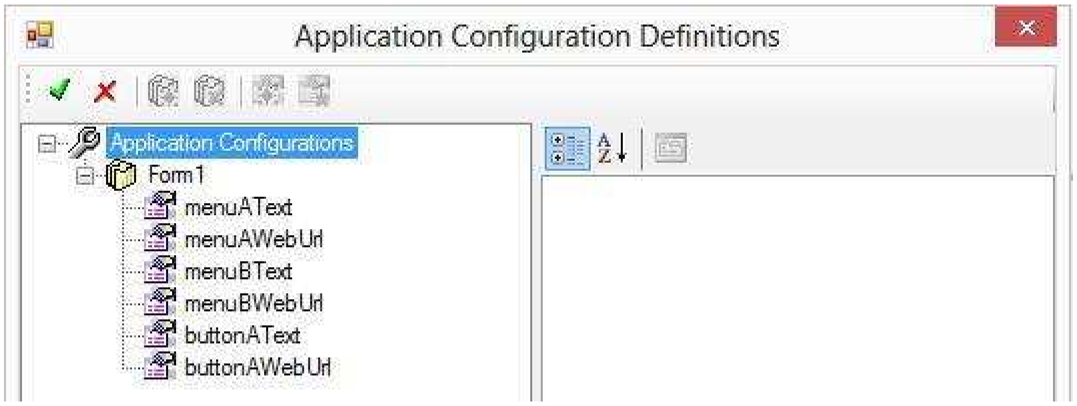
Click the menus and the button; we verify that they all function as if no branding is added.

## Set configuration values

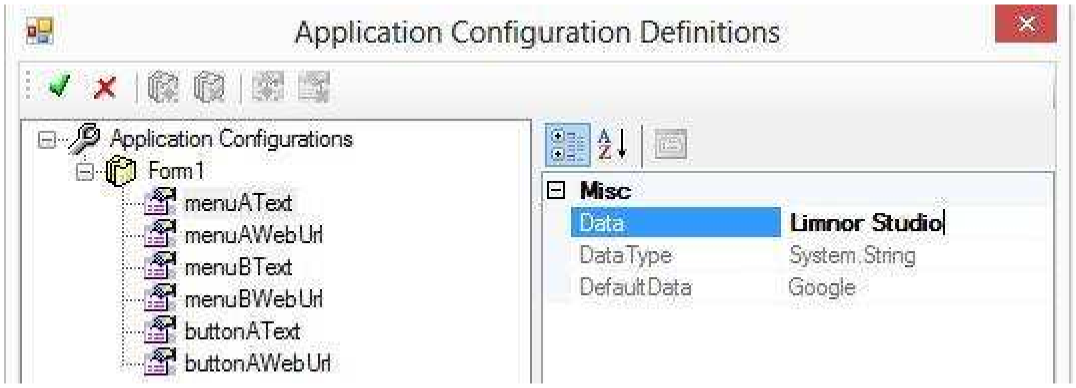
Now we choose menu "Branding" to set configuration values:



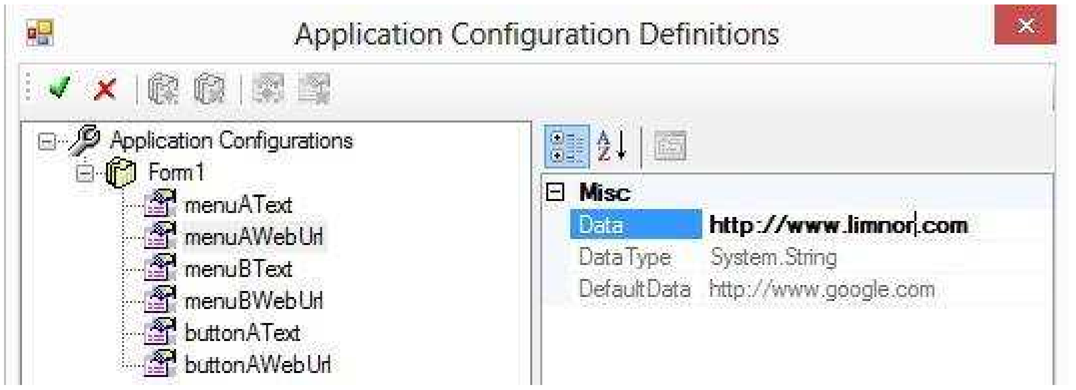
A dialogue box appears, allowing the user to modify configuration values:



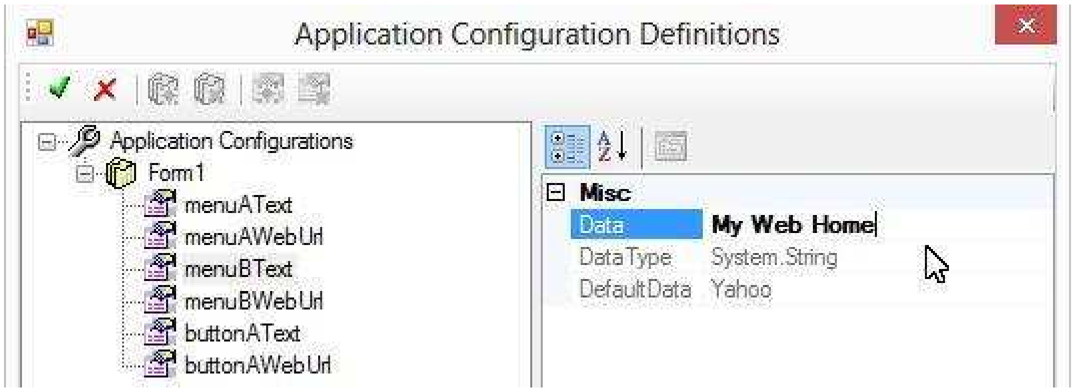
Modify configuration values:



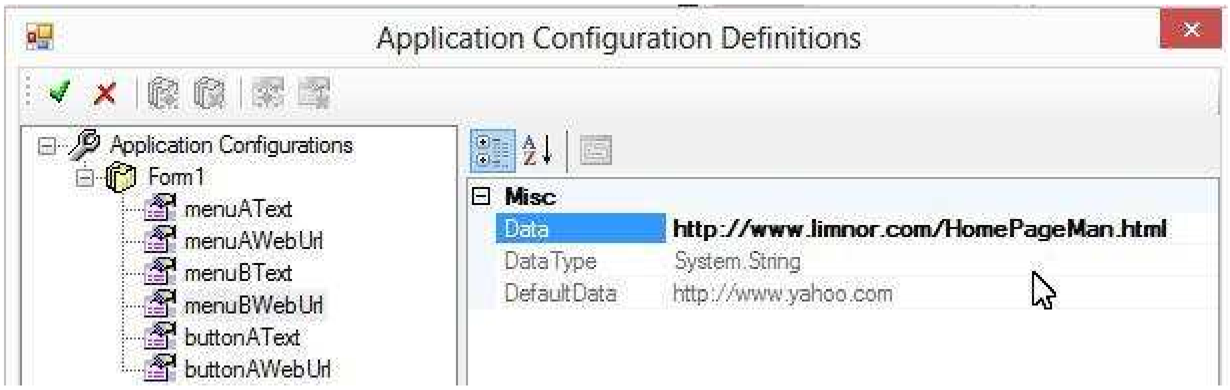
| | |
|---|---|
| Data | Limnor Studio |
| DataType | System.String |
| DefaultData | Google |



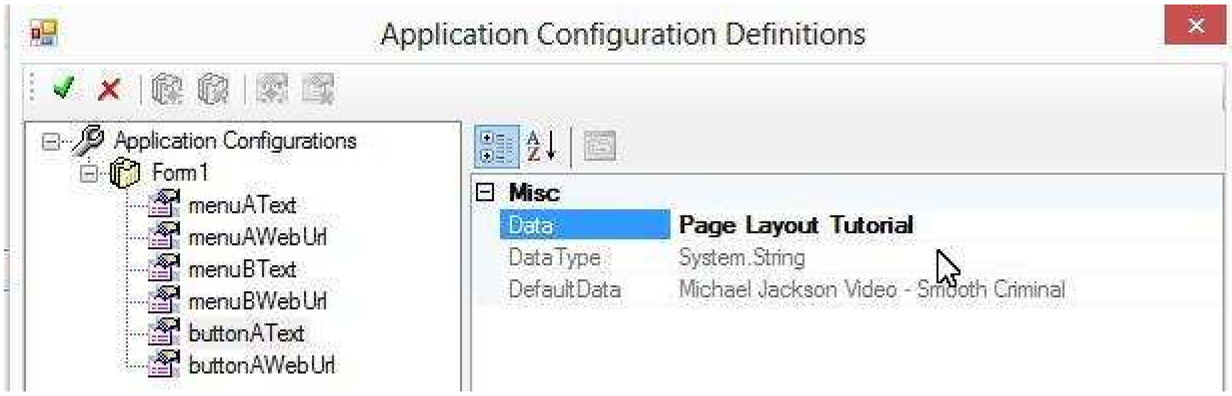
| | |
|---|---|
| Data | http://www.limnor.com |
| DataType | System.String |
| DefaultData | http://www.google.com |



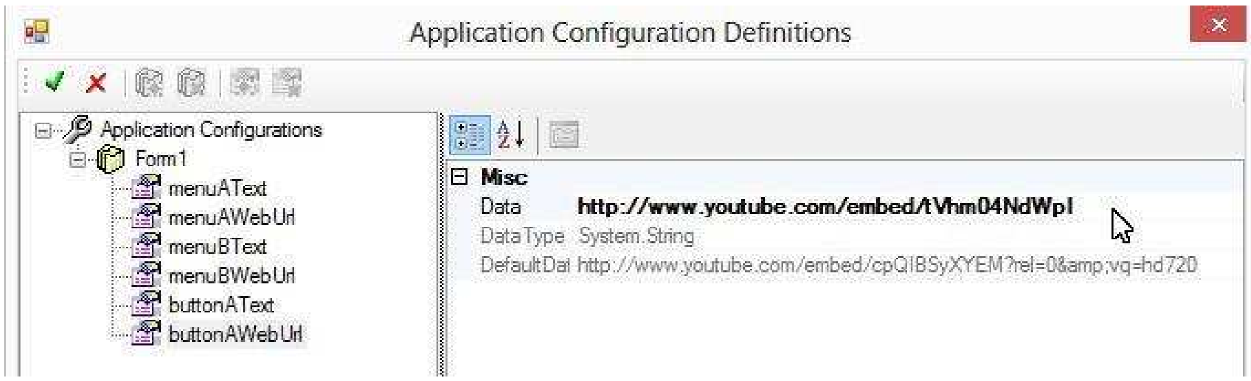
| | |
|---|---|
| Data | My Web Home |
| DataType | System.String |
| DefaultData | Yahoo |



| | |
|---|---|
| Data | http://www.limnor.com/HomePageMan.html |
| DataType | System.String |
| DefaultData | http://www.yahoo.com |

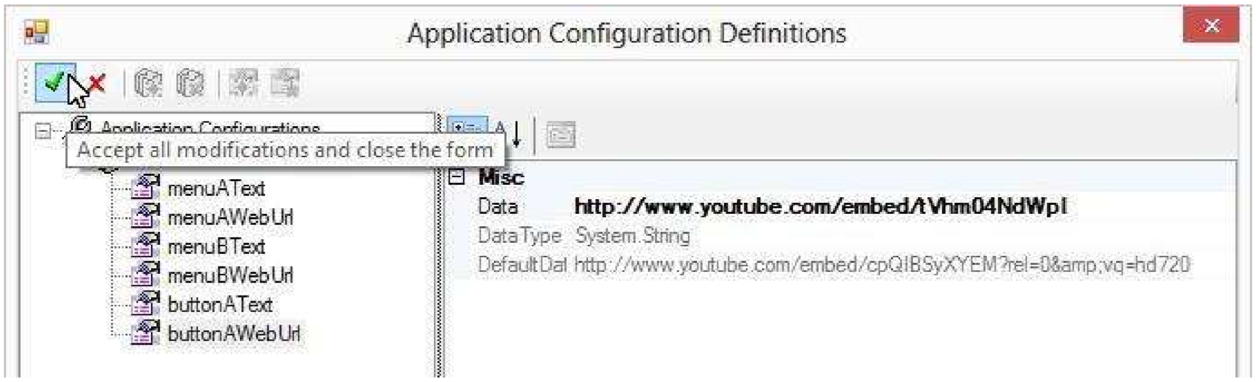


| | |
|---|---|
| Data | Page Layout Tutorial |
| DataType | System.String |
| DefaultData | Michael Jackson Video - Smooth Criminal |



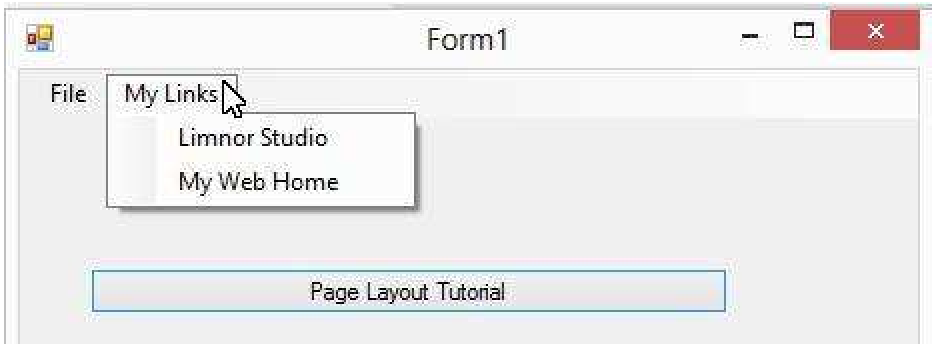
| | |
|---|---|
| Data | http://www.youtube.com/embed/tVhm04NdWpI |
| DataType | System.String |
| DefaultDat | http://www.youtube.com/embed/cpQIBSyXYEM?rel=0&amp;vq=hd720 |

Click the green check button to finish modifying the configuration values:


Application Configuration Definitions
Accept all modifications and close the form
Application Configurations
menuAText
menuAWebUrl
menuBText
menuBWebUrl
buttonAText
buttonAWebUrl
Misc

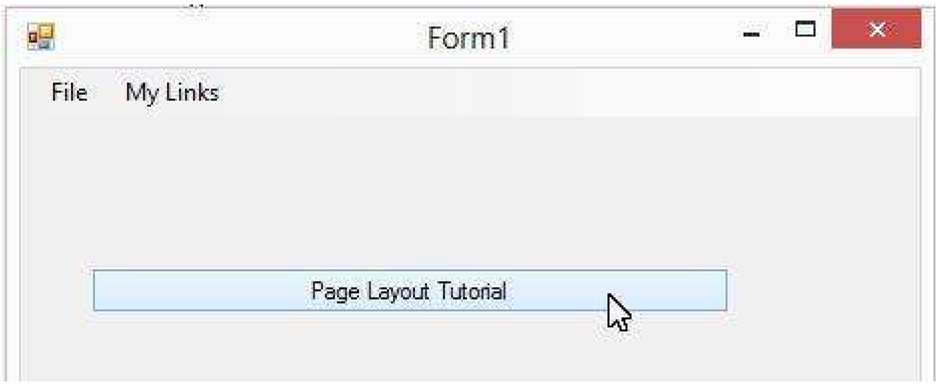
| | |
|---|---|
| Data | http://www.youtube.com/embed/tVhm04NdWpI |
| DataType | System.String |
| DefaultDat | http://www.youtube.com/embed/cpQiBSyXYEM?rel=0&amp;vq=hd720 |

Close Form1 and restart the program. Form1 appears. The components on the form change to the new configuration values:

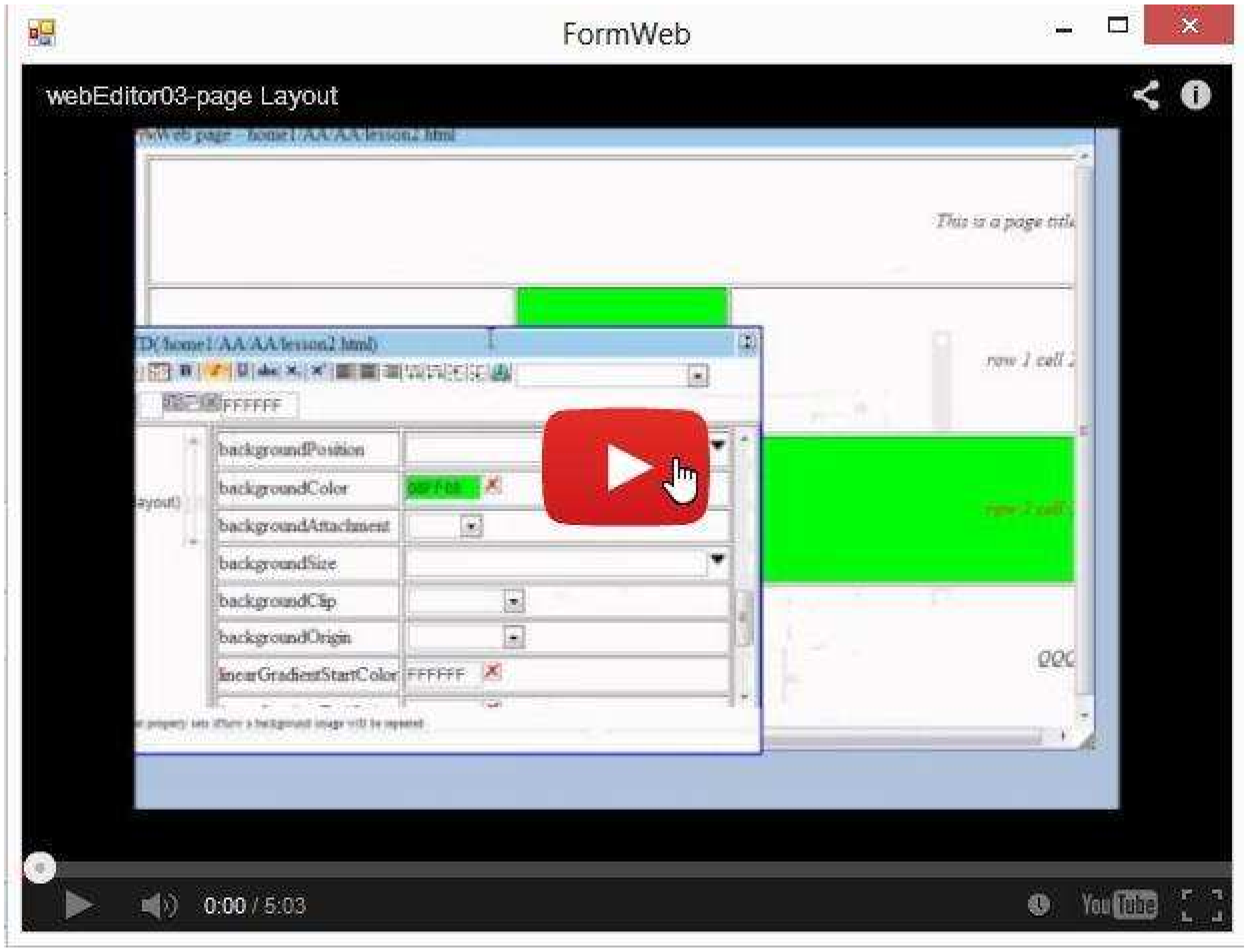


Choose menu "Limnor Studio", Limnor Studio web site appears, instead of Google web site.

Click the button:



Instead of the original Michael Jackson video, the page layout tutorial video appears:



We can see that the branding is weaved into our original software.

Note that once the above programming is done it is very easy to allow changing of other properties of the software. You just need to do 3 simple programming tasks to make a property changeable at runtime:

1. Create a new configuration value. See “Create vales” section on page 18.
2. Create an action to pass the new configuration value to the property to be made changeable.
3. Assign the new action to event LoadingConfigurations.

## Multiple brands

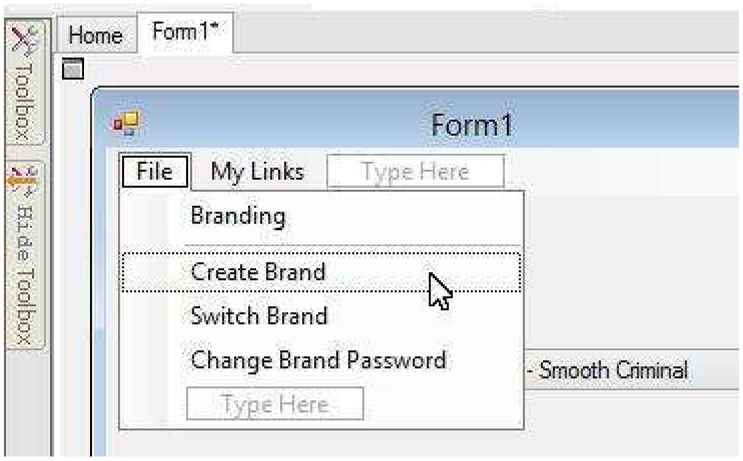
### Nameless Configurations

Multiple sets of configuration values may co-exist. The user may switch to different sets of configuration values. In the previous user test, we did not care about selecting a set of configuration values because we were using the “Nameless Configurations”. If you do not need to enable multiple set of configurations then your programming is done at this time.

For this sample, we’ll enable the use of configurations more than the nameless profile.

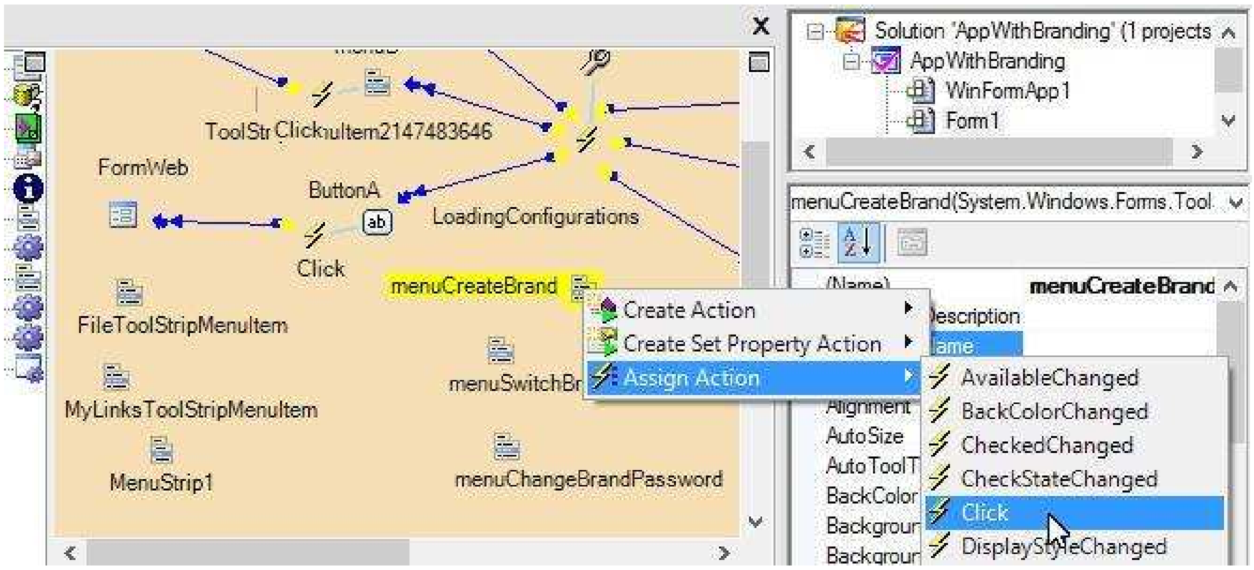
Because we are using configuration values for branding, we may call a set of configuration values a "brand".

In this sample, we use menus to allow the users to create and switch brands.

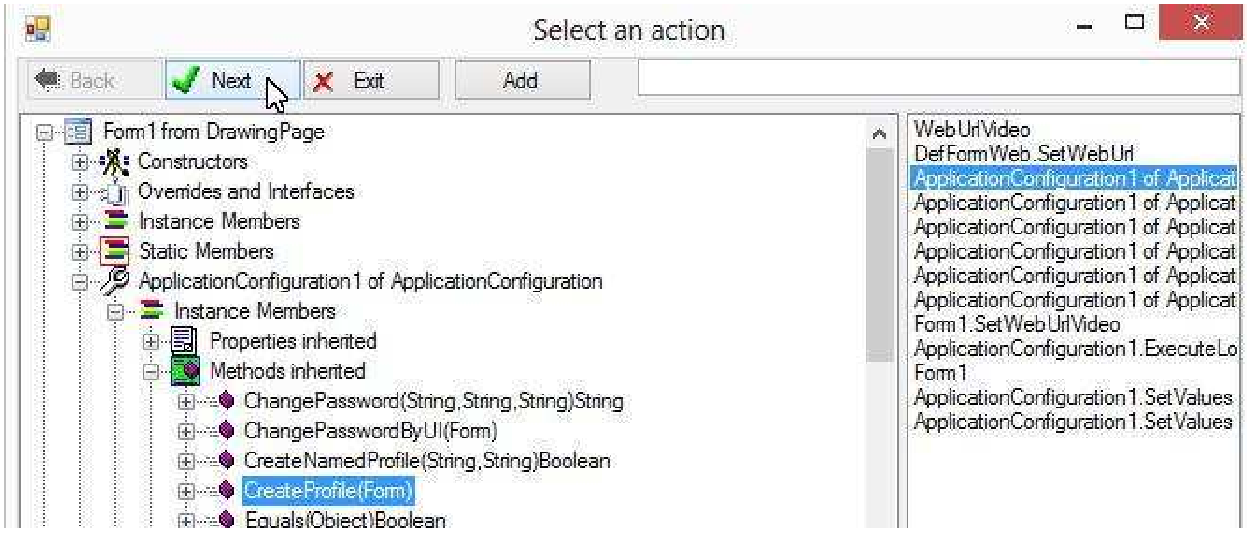


## Create Brand

Method CreateProfile of an Application Configuration component can be used to create new brands. Let's use a menu to execute CreateProfile.



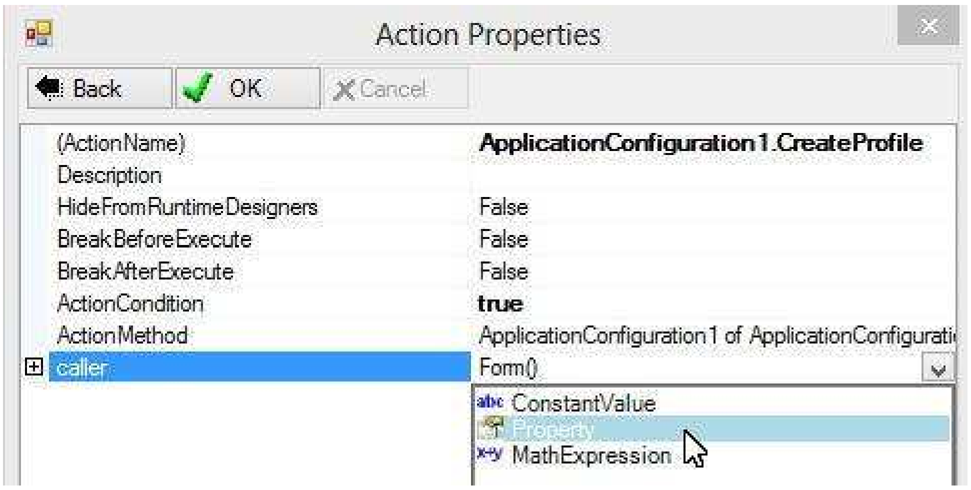
Select CreateProfile method of the Application Configuration component:



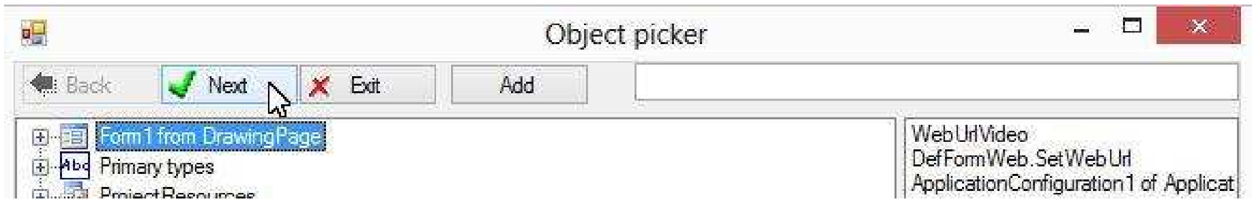
Form1 from DrawingPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
ApplicationConfiguration1 of ApplicationConfiguration
Instance Members
Properties inherited
Methods inherited
ChangePassword(String,String,String)String
ChangePasswordByUI(Form)
CreateNamedProfile(String,String)Boolean
CreateProfile(Form)
Equals(Object)Boolean

WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
Form1.SetWebUrlVideo
ApplicationConfiguration1.ExecuteLo
Form1
ApplicationConfiguration1.SetValues
ApplicationConfiguration1.SetValues

The action will launch a dialogue box. “caller” is the owner of the dialogue box. Use Form1 for it:



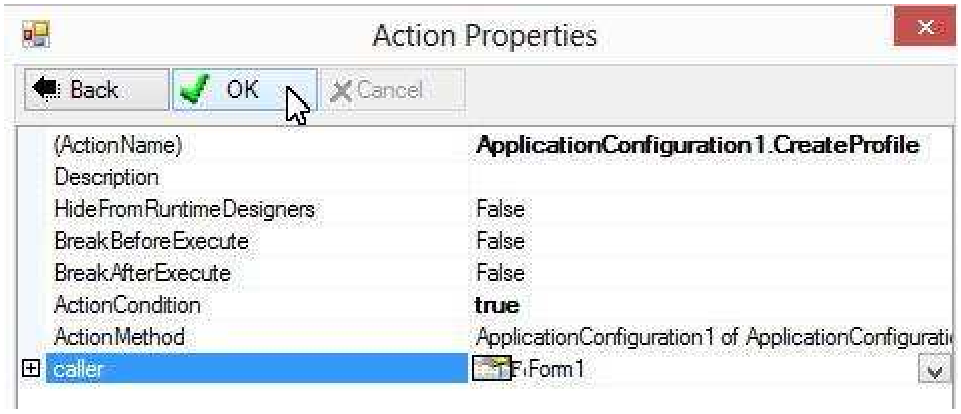
| | |
|---|---|
| (ActionName) | ApplicationConfiguration1.CreateProfile |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfigurati |
| caller | Form() |



Form1 from DrawingPage
Primary types
ProjectResources

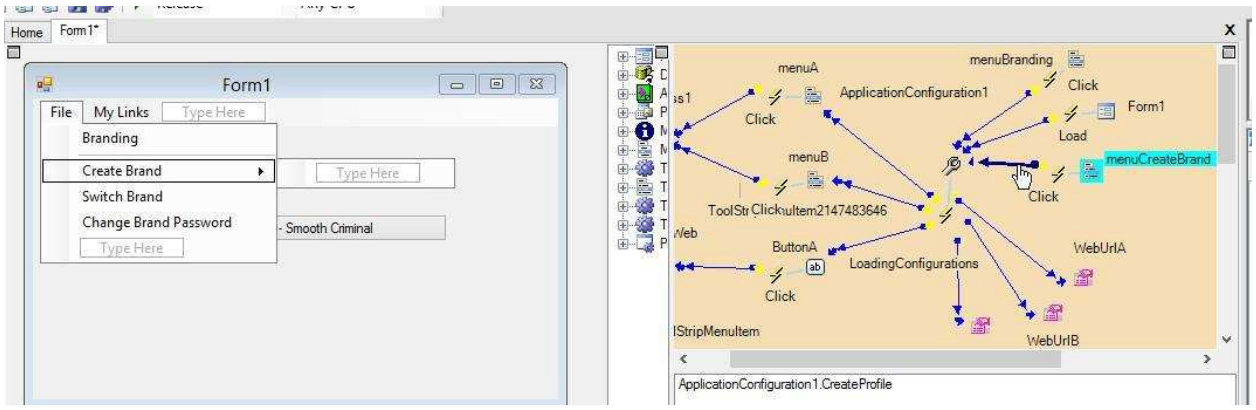
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat

Click OK:

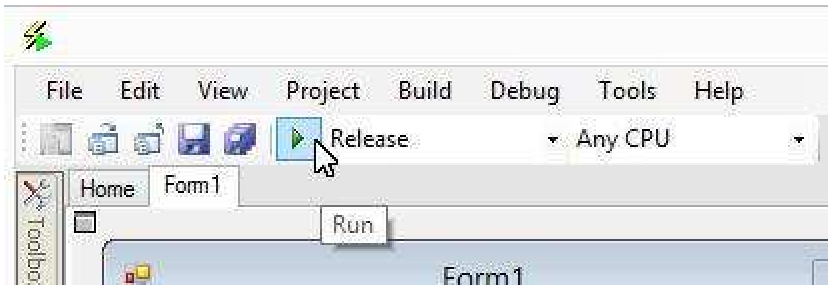


| | |
|---|---|
| (ActionName) | ApplicationConfiguration1.CreateProfile |
| Description | |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfigurati |
| caller | Form1 |

The action is created and assigned to the menu:

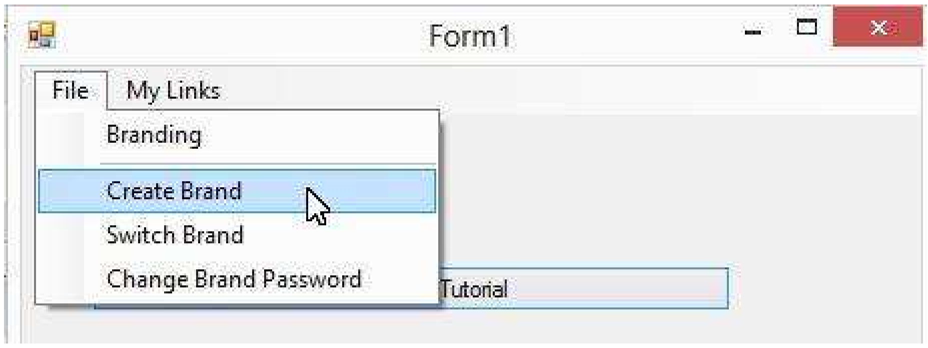


Brands are created at runtime by users. Let's run the program:

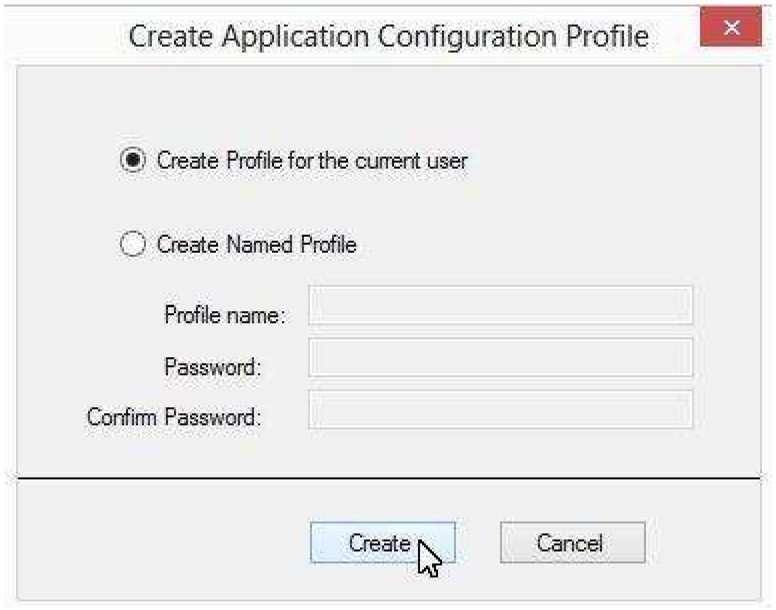


## Create user-specific brand

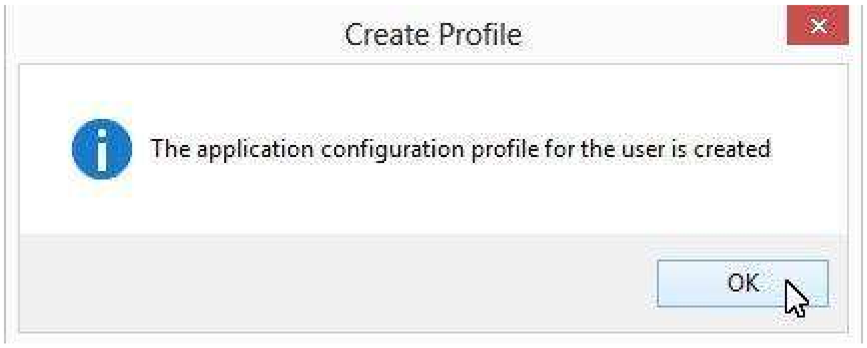
Choose menu "Create brand":



A dialogue box appears. While "Create Profile for the current user" is selected, click "Create" to create a user-specific brand:


Create Application Configuration Profile
Create Profile for the current user
Create Named Profile
Profile name:
Password:
Confirm Password:
Create
Cancel

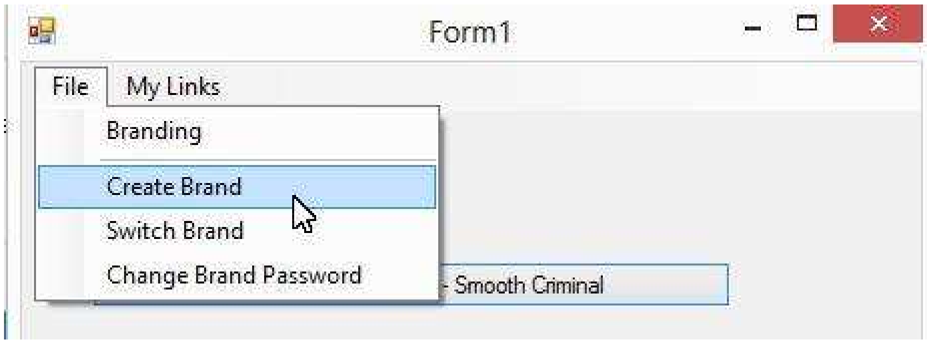
A message box appears:



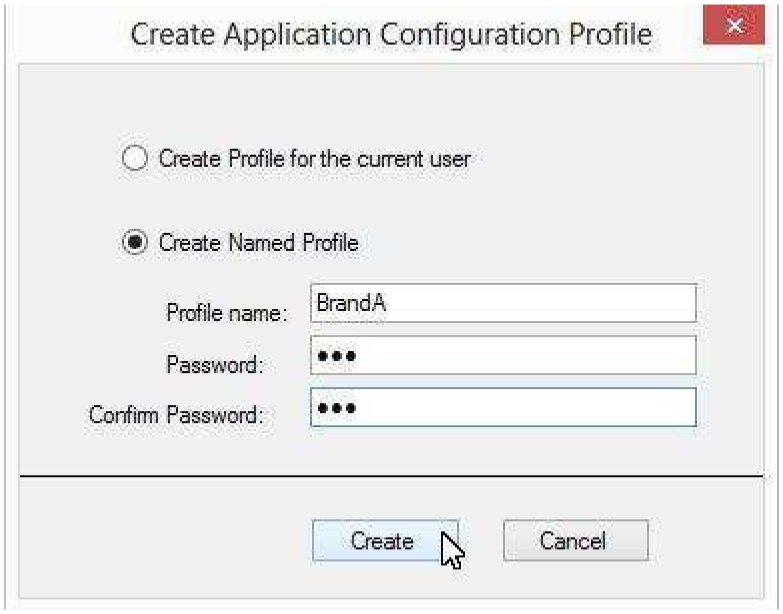
The application configuration profile for the user is created

## Create named brand

A user-specific brand can only be accessed if the same Windows user is using the computer. We may create named brands so that all users can access them. A named brand can also be password protected.

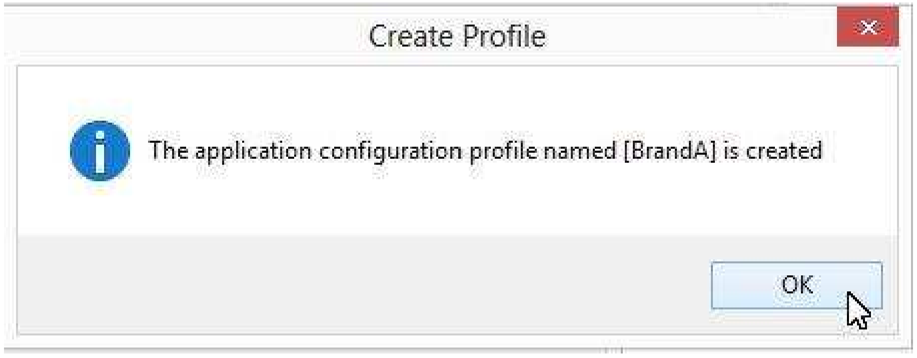


A named brand must be given a name:


Create Application Configuration Profile
Create Profile for the current user
Create Named Profile
Create
Cancel

Profile name: BrandA
Password: •••
Confirm Password: •••

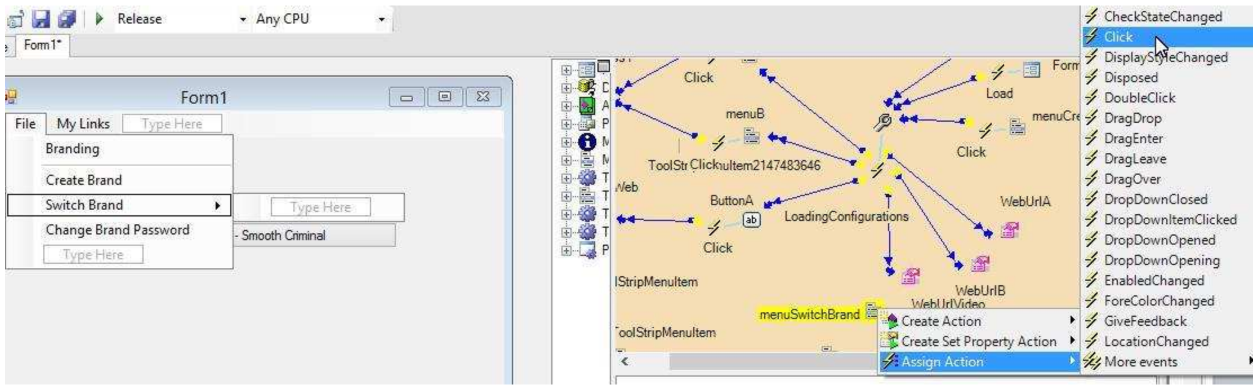
Click “Create”, a message box appears:



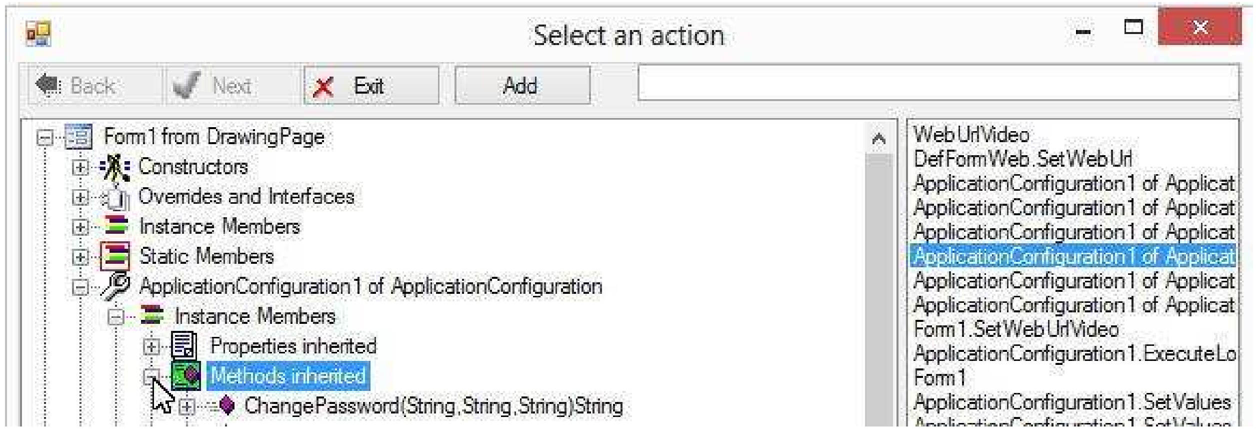
The application configuration profile named [BrandA] is created

## Switch brands

Method “LogOnProfileWithUI” of an Application Configuration component can be used to switch brand. Let’s use a menu to invoke it:

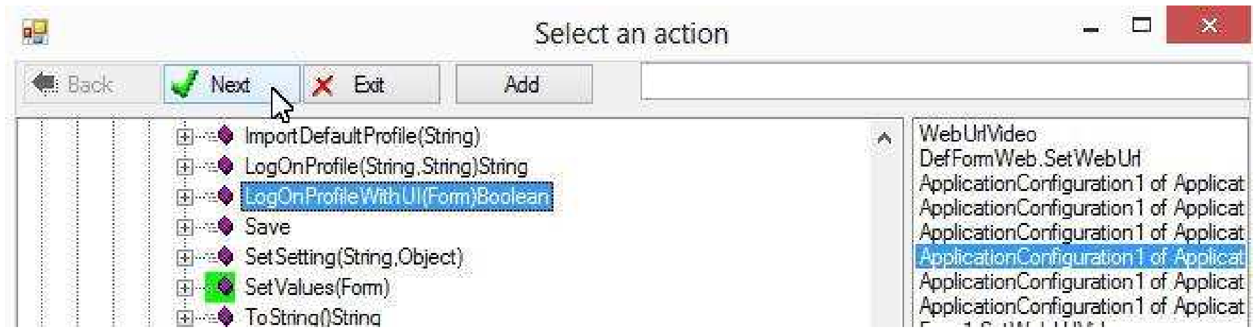


Choose LogOnProfileWithUI method of the Application Configuration component:



Form1 from DrawingPage
Constructors
Overrides and Interfaces
Instance Members
Static Members
ApplicationConfiguration1 of ApplicationConfiguration
Instance Members
Properties inherited
Methods inherited
ChangePassword(String,String,String)String

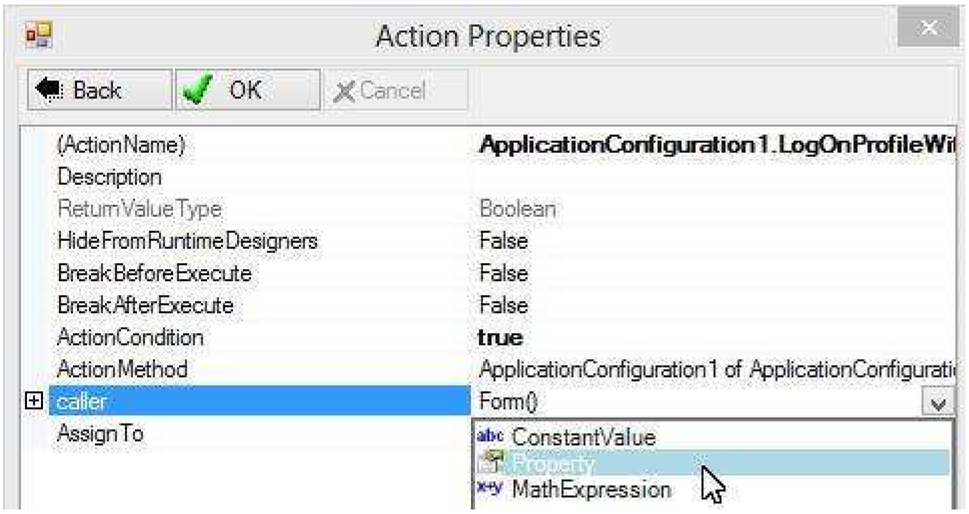
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
Form1.SetWebUrlVideo
ApplicationConfiguration1.ExecuteLo
Form1
ApplicationConfiguration1.SetValues



ImportDefaultProfile(String)
LogOnProfile(String,String)String
LogOnProfileWithUI(Form)Boolean
Save
SetSetting(String,Object)
SetValues(Form)
ToString()String

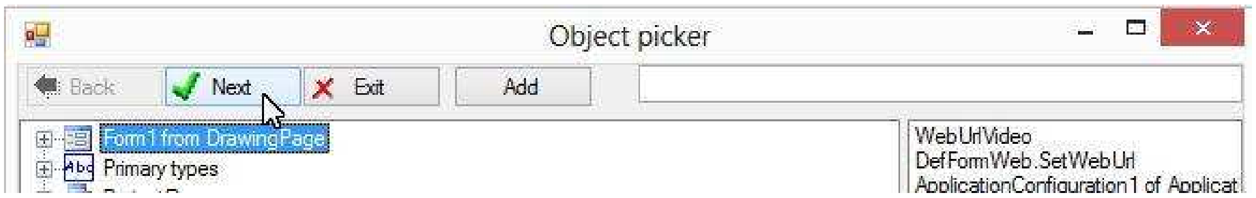
WebUrlVideo
DefFormWeb.SetWebUrl
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat
ApplicationConfiguration1 of Applicat

Set “caller” to Form1:



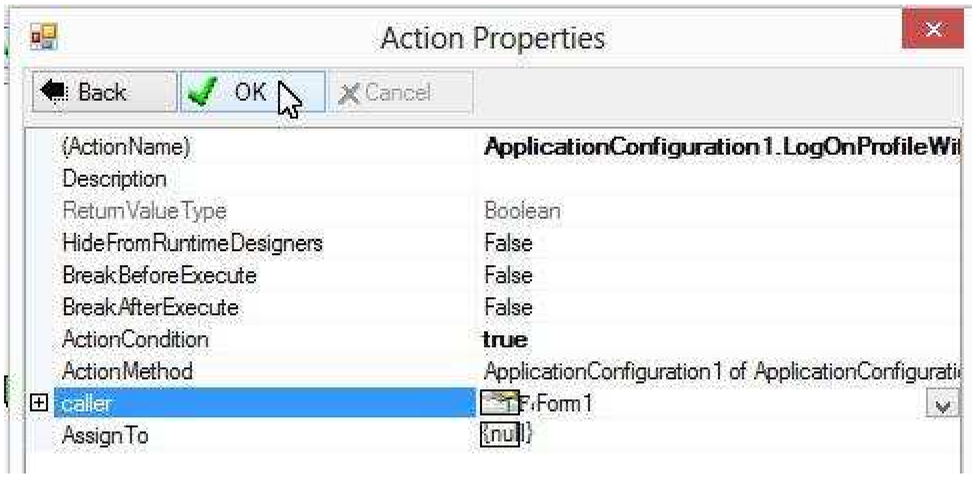
Action Properties

|  |  |
|---|---|
| (ActionName) | ApplicationConfiguration1.LogOnProfileWi |
| Description | |
| ReturnValueType | Boolean |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfigurati |
| caller | Form() |
| AssignTo | |



Object picker

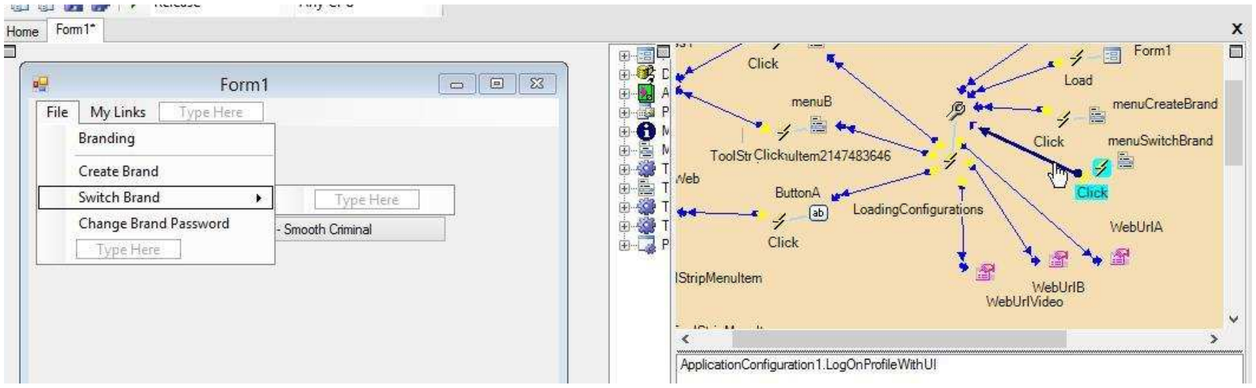
Click OK:



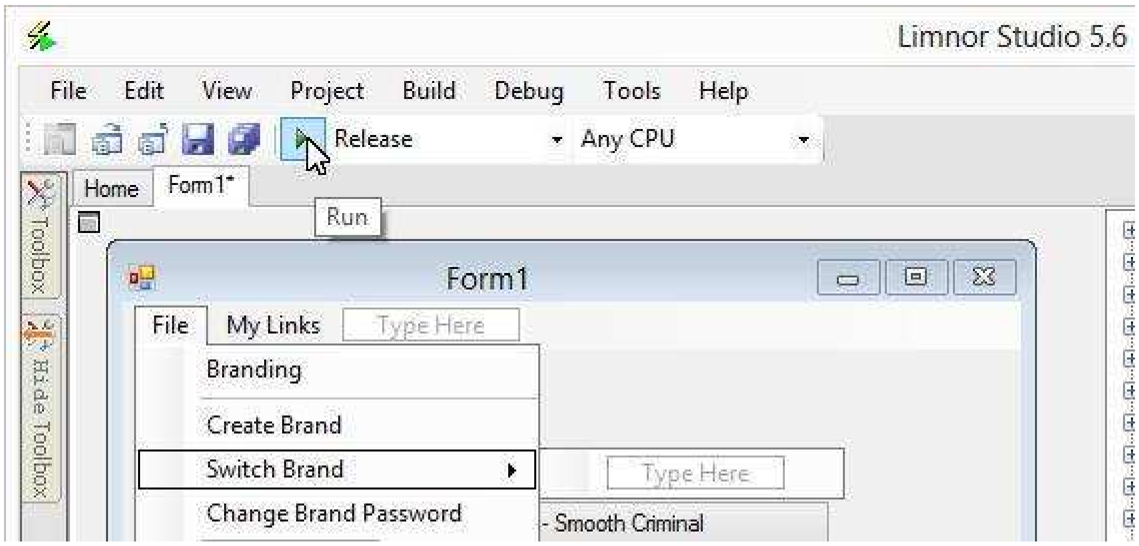
Action Properties

|  |  |
|---|---|
| (ActionName) | ApplicationConfiguration1.LogOnProfileWi |
| Description | |
| ReturnValueType | Boolean |
| HideFromRuntimeDesigners | False |
| BreakBeforeExecute | False |
| BreakAfterExecute | False |
| ActionCondition | true |
| ActionMethod | ApplicationConfiguration1 of ApplicationConfigurati |
| caller | F.Form1 |
| AssignTo | {null} |

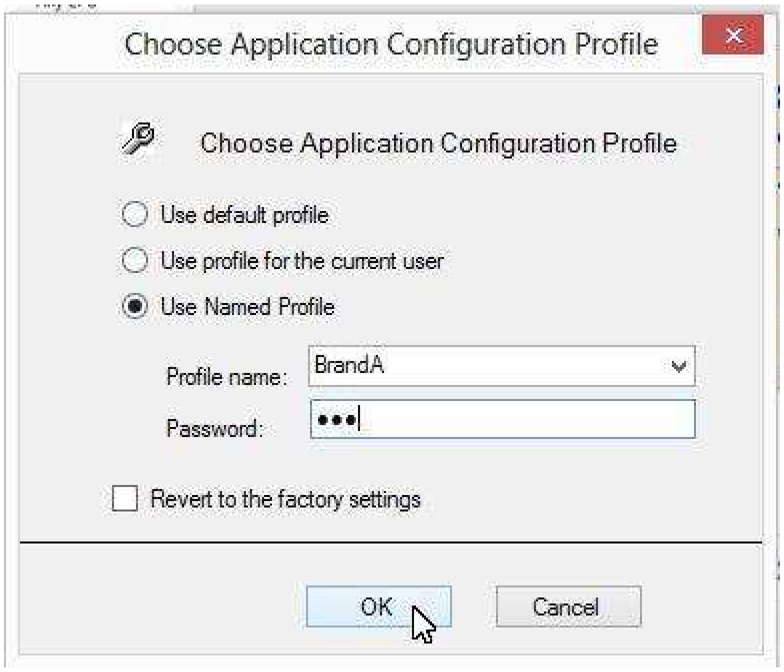
The action is created and assigned to the menu:



Let’s run the program to try to switch brand:



A dialogue box appears to let the user select a set of configuration values. The system calls a set of configuration values a “profile”. In our branding sample, a set of configuration values is a “brand”.

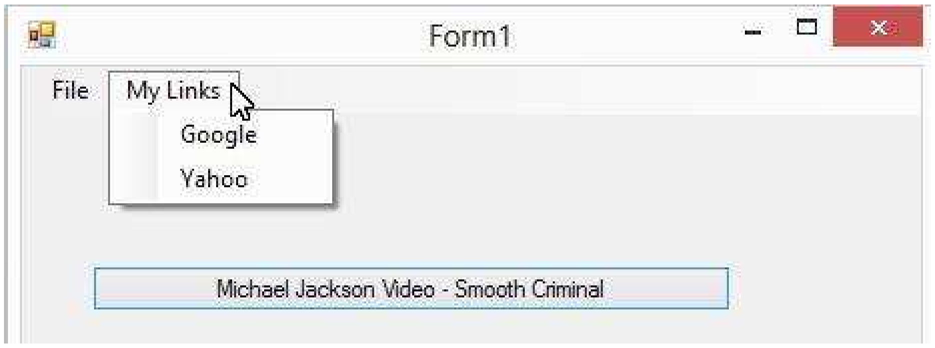

Choose Application Configuration Profile
Choose Application Configuration Profile
Use default profile
Use profile for the current user
Use Named Profile
Revert to the factory settings
OK
Cancel

Profile name: BrandA
Password: •••

The user has 3 options in selecting a brand:

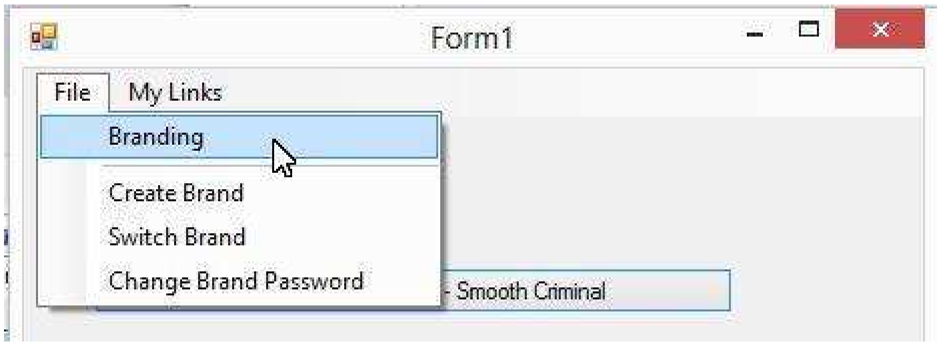
- Use default profile – this is the nameless profile or the set of configurations distributed with the program if there is not a nameless profile.
- Use profile for the current user – this is the set of configurations created by the current Windows user.
- Use Named Profile – select from named profiles created by users previously.

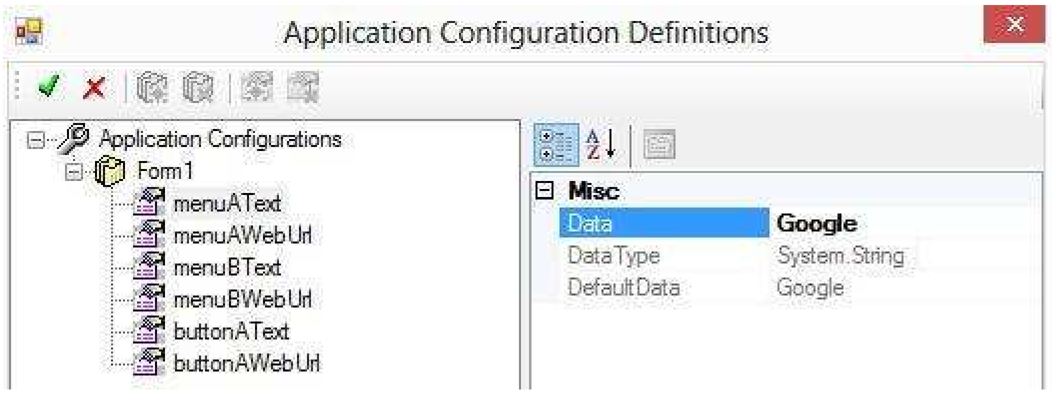
Click OK, Form1 appears:



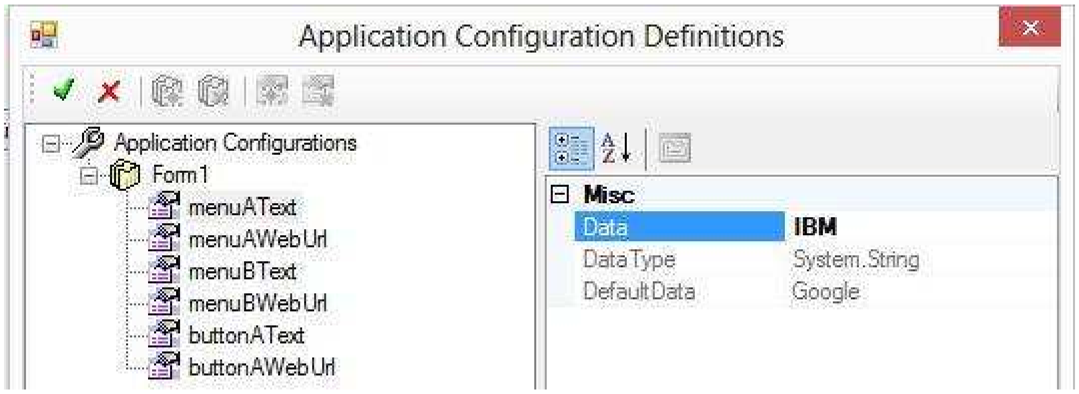
Note that all components revert to its original settings.

At this time, if the user makes modifications then modifications will be made to the current brand:

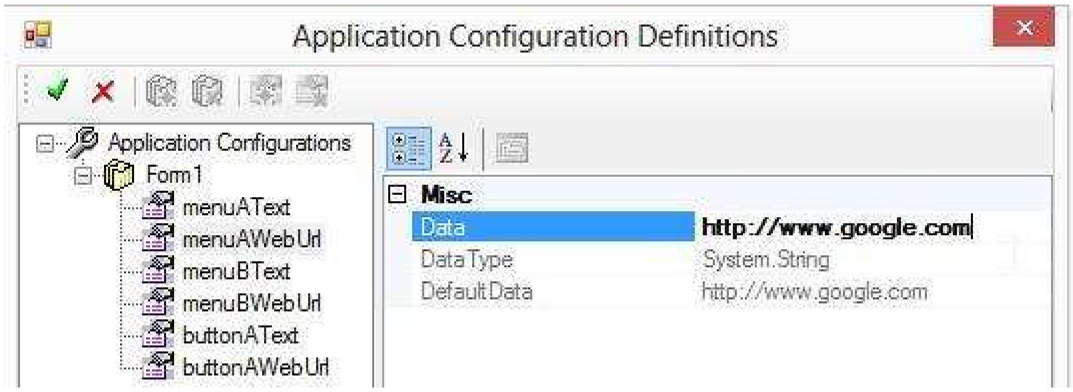




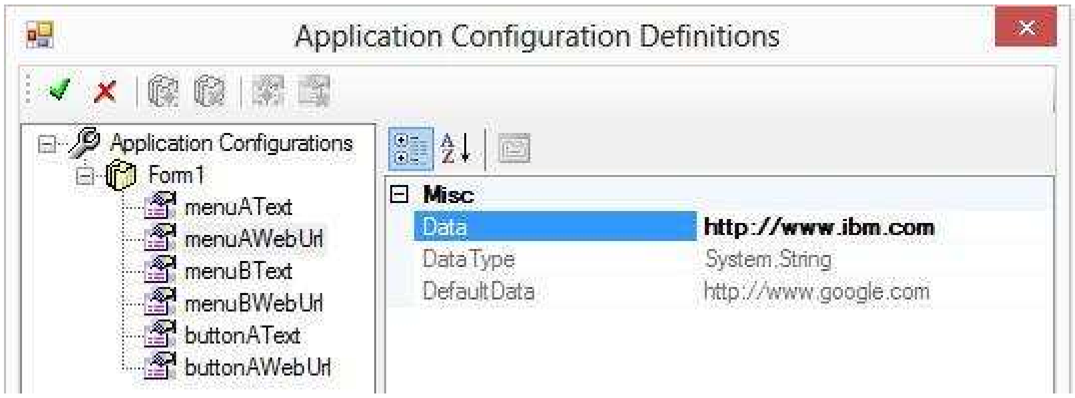
| | |
|---|---|
| Data | Google |
| DataType | System.String |
| DefaultData | Google |



| | |
|---|---|
| Data | IBM |
| DataType | System.String |
| DefaultData | Google |

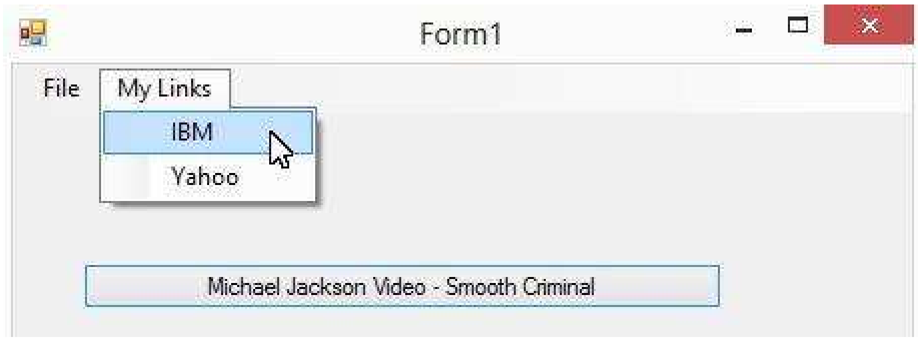


| | |
|---|---|
| Data | http://www.google.com |
| DataType | System.String |
| DefaultData | http://www.google.com |

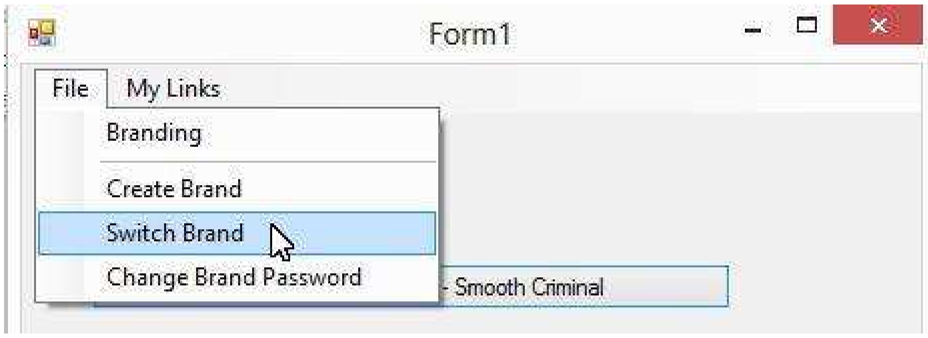


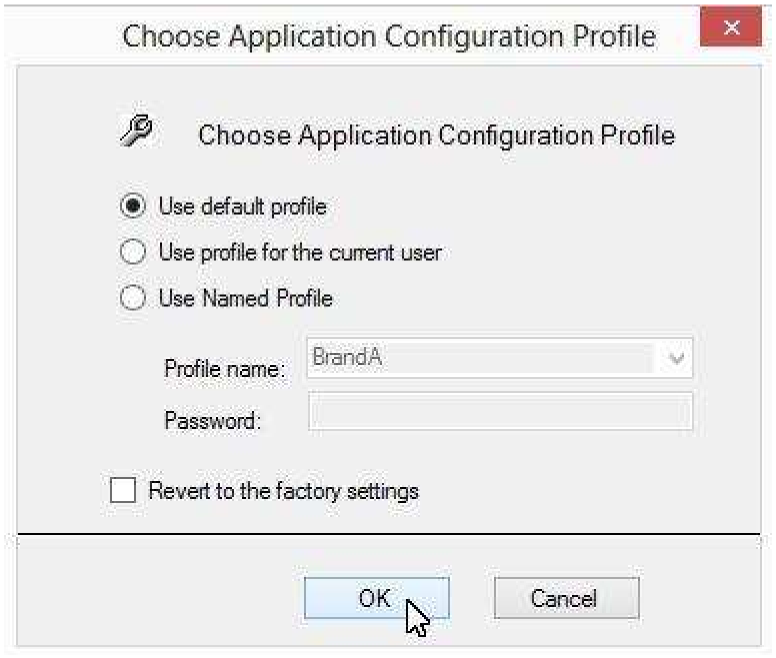
| | |
|---|---|
| Data | http://www.ibm.com |
| DataType | System.String |
| DefaultData | http://www.google.com |

Restart the program; select the same brand; the branding is applied:

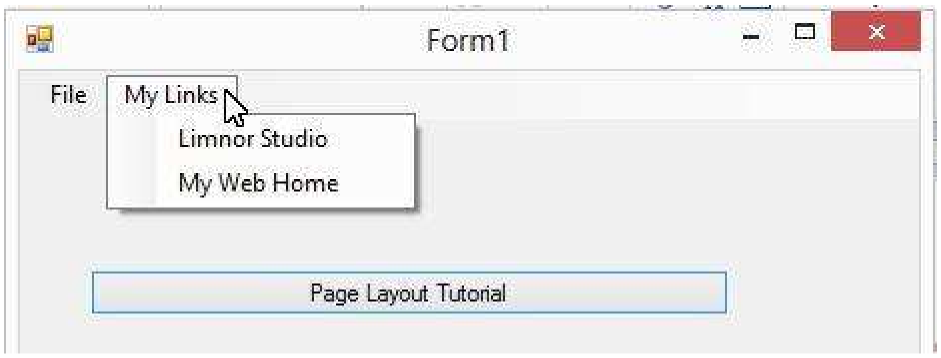


Let’s switch to the default profile:




Choose Application Configuration Profile
Choose Application Configuration Profile
● Use default profile
○ Use profile for the current user
○ Use Named Profile
Profile name: BrandA
Password:
☐ Revert to the factory settings
OK
Cancel

We can see that our previous settings are not affected:



## Configuration File Locations

Configuration values are saved in configuration files. The location of a configuration file depends on the type of the profile.

### Factory settings

Factory settings are saved in a file name as {EXE file name}.cfg. For example, suppose {EXE file name} is AppWithBranding.EXE then the file name is AppWithBranding.EXE.cfg.

This is how you may distribute desired default settings to the end users: copy a configuration file to the distribution folder and name it as {EXE file name}.cfg. The function “CopyProfile” helps you do the copying.

### Nameless profile

The default profile (factory settings) is saved in file

{ProgramData}\Limnor Studio\App Config\{EXE file name}_{app key}_.cfg

For example, suppose on your computer, {ProgramData} is C:\ProgramData, your {EXE file name} is AppWithBranding.EXE, the {app key} is 6bab1106b00f4da39345524722f98aae, and then the default settings are saved in the following file:

C:\ProgramData\Limnor Studio\App Config\AppWithBranding.EXE_6bab1106b00f4da39345524722f98aae_.cfg

### Named profile

The configurations file for a named profile is in the same folder as the default profile except that the file is named as {EXE file name}_{app key}_{name}.cfg, where {name} is the profile name.

### User profile

The configuration file for a user profile is saved in {Users}\{user name}\AppData\Local\{EXE file name}_{app key}.cfg

For example, if on your computer, {Users} is C:\Users, your Windows log on name is Admin, your {EXE file name} is AppWithBranding.EXE, the {app key} is 6bab1106b00f4da39345524722f98aae, and then the configuration file is

C:\Users\Admin\AppData\Local\AppWithBranding.EXE_6bab1106b00f4da39345524722f98aae.cfg

### Distribution with desired factory settings

The above information is for your reference. You do not have to remember the information. Function CopyProfile automatically gets the current profile and copy it to the location the user specifies and name it to {EXE file name}.cfg. See http://www.limnor.com/support/CreateAppConfigUtility.pdf for an example of using CopyProfile.

## Feedback

Please send your feedback and suggestions to support@limnor.com