



FoxInCloud

aw.vcx at a glance

Properties & Methods that adapted classes and objects inherit from aw.vcx

AT20 Development Workshop – Day 1, Activity 6



Agenda

Section	Subject	Duration
1	Architecture	10'
2	Common PMs	2'
3	aw.vcx!awFrm as Form	10'
4	aw.vcx!awGrd as Grid	2'
5	Other classes	2'



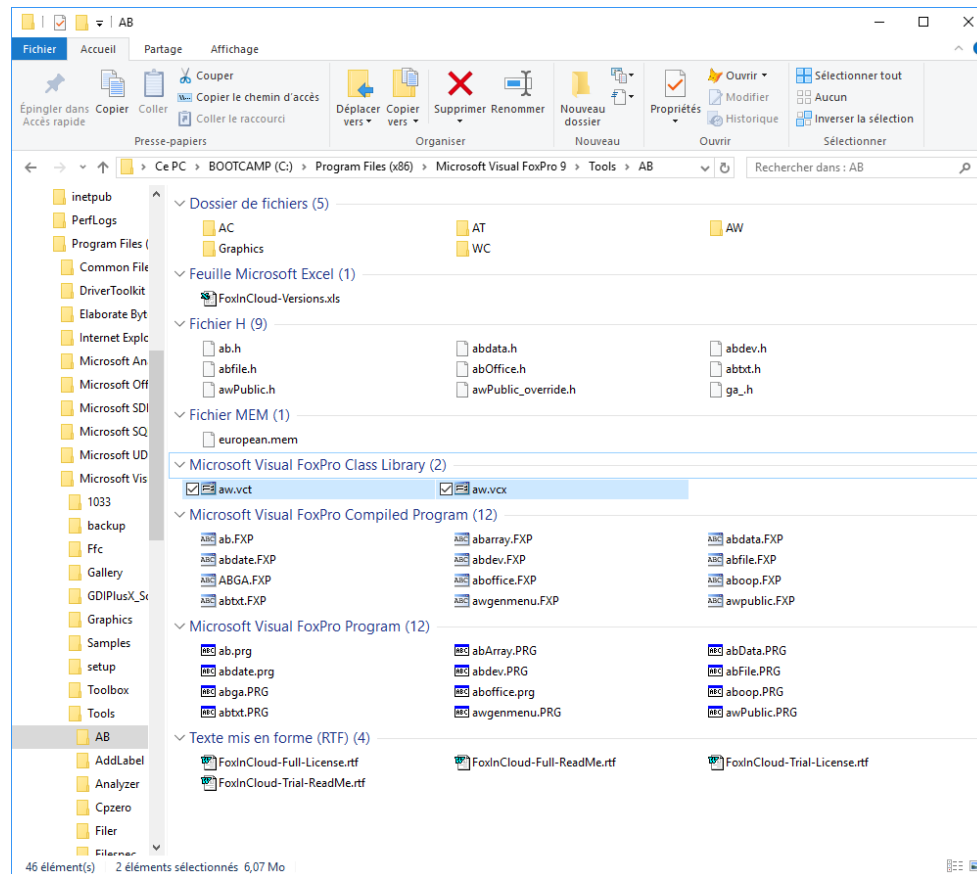
Installed stuff

FAA and FAS install aw.vcx and dependencies (AKA 'FoxInCloud Public Layer') into:

Home (1) +
'tools\ab\'

'AB' stands for
'Abaque' (com
pany behind
FiC)

'AW' stands for
'Abaque Web'



AB: general
purpose, public

AC: general
purpose,
licensed

AT: dev tools,
public

AW: web-
specific,
licensed

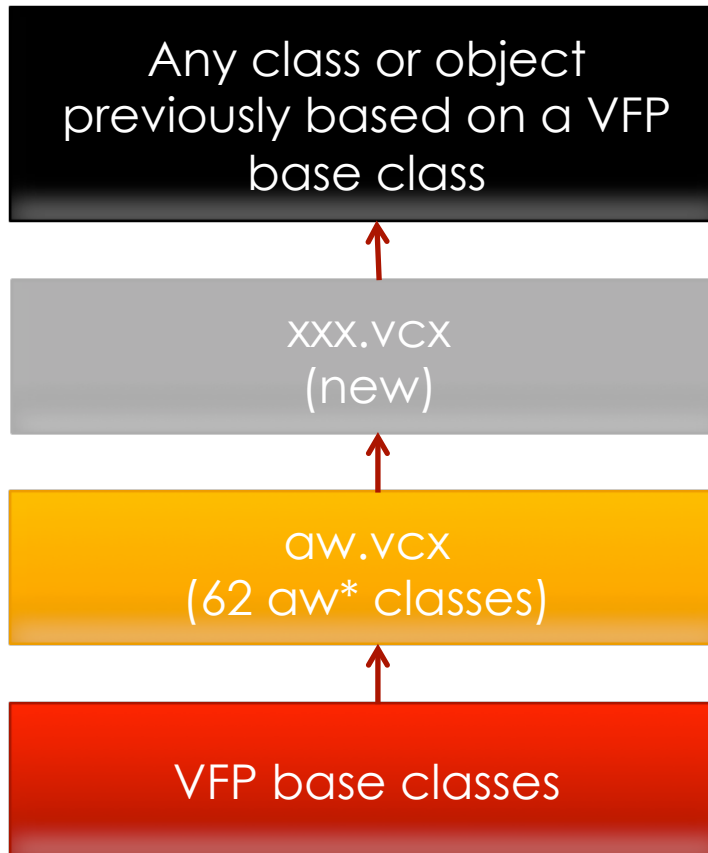
WC: West-Wind
Web connect,
licensed



Class inheritance after adaptation

After FAA step 2, each class and object in the project inherits from aw.vcx, the FoxInCloud 'base' class library.

FAA creates an intermediary class library for you: xxx.vcx



Your application classes and objects

FAA never overwrites – implement application wide code here

Can change at any time; never modify!



FoxInCloud naming conventions are designed to be as familiar as possible while limiting the risk of conflict with your naming scheme

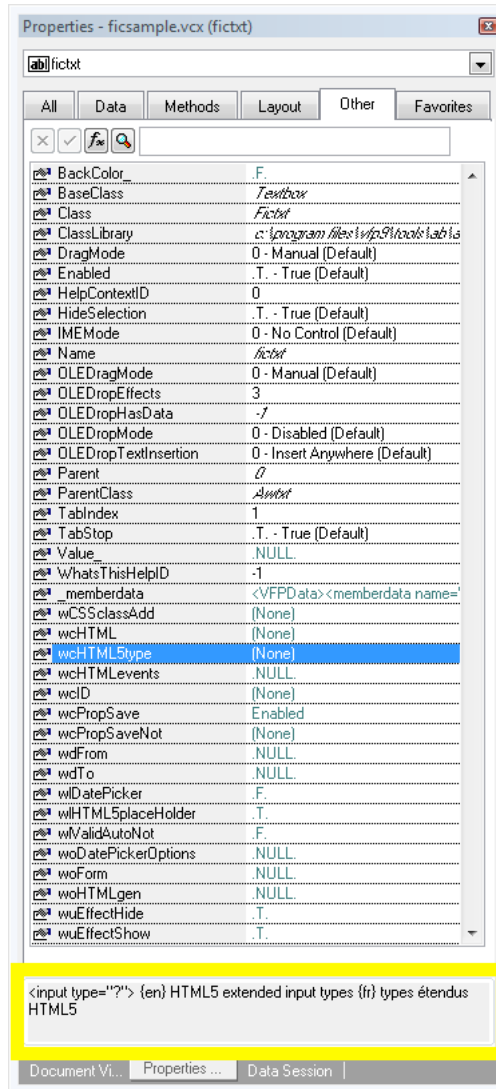
Naming conventions

- Classes: follows “Object Naming Conventions” as of VFP help; eg.
FoxInCloud base `Textbox` class: `awTxt`
 - Properties, Methods, Procedures follow ‘object oriented’ naming:
`[w] [type]Element [Attribute] [Action]`
 - `[w]` denotes a web-specific PM
 - `[type]` is a standard type prefix, eg. ‘c’ for character
- eg.
- Show Form: `wFormShow()`
 - Set Session Variable: `wSessionVarSet()`
 - Properties to be saved: `wcPropSave`

Documentation



You can get documentation right in your VFP IDE



Properties - ficsample.vcx (fictxt)

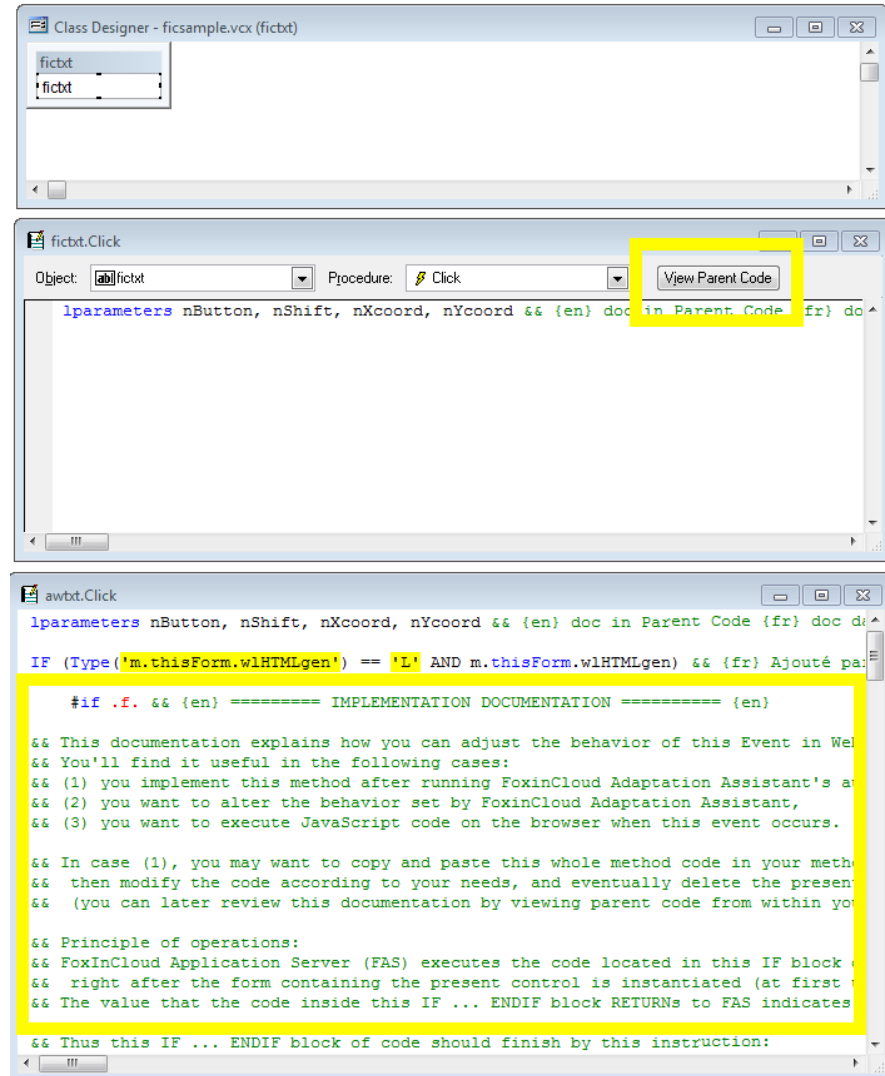
ab|fictxt

All Data Methods Layout Other Favorites

BackColor	.F.
BaseClass	Textbox
Class	Fictxt
ClassLibrary	c:\program files\microsoft\visual foxpro\bin\src\lib\lib
DragMode	0 - Manual (Default)
Enabled	.T. - True (Default)
HelpContextID	0
HideSelection	.T. - True (Default)
IMEMode	0 - No Control (Default)
Name	fictxt
OLEDragMode	0 - Manual (Default)
OLEDropEffects	3
OLEDropHasData	-1
OLEDropMode	0 - Disabled (Default)
OLEDropTextInsertion	0 - Insert Anywhere (Default)
Parent	0
ParentClass	Awtxt
TabIndex	1
TabStop	.T. - True (Default)
Value	.NULL.
WhatsThisHelpID	-1
memberdata	<VFP data> <memberdata name=
wCSSclassAdd	(None)
wcHTML	(None)
wcHTML5type	(None)
wcHTML5events	.NULL.
wcID	(None)
wcPropSave	Enabled
wcPropSaveNot	(None)
wdFrom	.NULL.
wdTo	.NULL.
wDatePicker	.F.
wHTML5placeHolder	.T.
wValidAutoNot	.F.
woDatePickerOptions	.NULL.
wofForm	.NULL.
woHTMLgen	.NULL.
wuEffectHide	.T.
wuEffectShow	.T.

<input type="?"> {en} HTML5 extended input types {fr} types étendus HTML5

Document Vi... Properties ... Data Session |



Class Designer - ficsample.vcx (fictxt)

fictxt

fictxt

fictxt.Click

Object: ab|fictxt Procedure: Click View Parent Code

```
lparameters nButton, nShift, nXcoord, nYcoord && {en} doc in Parent Code {fr} do
```

awtxt.Click

```
lparameters nButton, nShift, nXcoord, nYcoord && {en} doc in Parent Code {fr} doc de
```

```
IF (Type('m.thisForm.wlHTMLgen') == 'L' AND m.thisForm.wlHTMLgen) && {fr} Ajouté pa
```

```
#if .F. && {en} ===== IMPLEMENTATION DOCUMENTATION ===== {en}
```

```
&& This documentation explains how you can adjust the behavior of this Event in We
```

```
&& You'll find it useful in the following cases:
```

```
&& (1) you implement this method after running FoxInCloud Adaptation Assistant's a
```

```
&& (2) you want to alter the behavior set by FoxInCloud Adaptation Assistant,
```

```
&& (3) you want to execute JavaScript code on the browser when this event occurs.
```

```
&& In case (1), you may want to copy and paste this whole method code in your meth
```

```
&& then modify the code according to your needs, and eventually delete the presen
```

```
&& (you can later review this documentation by viewing parent code from within yo
```

```
&& Principle of operations:
```

```
&& FoxInCloud Application Server (FAS) executes the code located in this IF block
```

```
&& right after the form containing the present control is instantiated (at first
```

```
&& The value that the code inside this IF ... ENDIF block RETURNS to FAS indicates
```

```
&& Thus this IF ... ENDIF block of code should finish by this instruction:
```



Desktop app distribution

All classes and procedures dependencies that FAA add to your app. are under GNU General Public License.

Free to use and compile into your desktop exe.

- You can take advantage of 1,000 + General Purpose Modules in 10 `ab*.prg`: `abArray`, `abTxt`, `abDev`, `abFile`, `abData`, etc.
- Automatic update by any new version of FAA or FAS
- FAS install grants rights to all users on `home (1) + 'tools\ab\'` but not FAA – **make sure not to modify code or run FAA/VFP as an Administrator**
- FoxInCloud Public Layer on [GitHub](https://github.com/FoxInCloud/FoxInCloud-AB):
<https://github.com/FoxInCloud/FoxInCloud-AB>
- Want to improve code? Fork on [Github](https://github.com)!



aw.vcx: common PMs

1- properties

Property	Usage
<code>.wcPropSave</code>	Properties defining the user state of a control (user-defined properties added automatically)
<code>.wcID (read-only)</code>	HTML unique ID for the control; for use in CSS and/or JavaScript
<code>.wuEffect*</code>	Defines whether an effect is played when control is shown or hidden in browser
<code>.wCSSClassAdd</code>	CSS classes to be added to the rendered HTML
<code>.wlContentDynamic</code>	User event code can add/remove members to container object



aw.vcx: common PMs 2- methods

Methods described here work in either desktop mode or Web mode: one single call for either mode.

Method	Usage
<code>.wFormCallback* ()</code>	Empty methods where you can move your call back code; this standard name helps finding the call back code; however you can locate this code in any method of your choice.
<code>.wcHTMLgen ()</code>	Implement this method if you need custom HTML for a control.
<code>.wReadMe ()</code>	Class documentation



aw.vcx!awFrm as Form 1- properties

Most of the useful PMs are gathered in this class ... make sure to get familiar with them!

Property	Usage
<code>.wlWeb, .wlLAN</code>	Indicates whether form runs in web or desktop mode; useful to bracket code for either mode.
<code>.wlInitFirst</code>	Form.Init() runs at initial form instantiation, out of user context
<code>.wcModalChoiceProp</code>	For modal forms returning a value, name of thisForm.property where this value is stored (default is <code>'wuValue'</code>)
<code>.wcWindowTheme</code>	Theme for styling the Form Window in Web mode
<code>.wlAnonymousAllowed</code>	Anonymous users can access this form (see <code>.wUserLogin()</code> below)



aw.vcx!awFrm as Form 2- methods

Methods described here work in either desktop mode or Web mode: one single call for either mode.

Method	Usage
<code>.wForm* ()</code>	Opens a form.scx or class and shows it immediately or later; form is modal if <code>.WindowType=1</code> and <code>isNull (parm2)</code> or <code>parm2</code> is not empty or specifies a callback method.
<code>.wMessageBox ()</code> <code>.wInputBox ()</code>	Opens a message / input box using the standard VFP parameters, optional call back method name as parm # 1
<code>.wFileSaveAs ()</code>	Displays a 'save file' dialog in either modes; useful for exports and PDF reporting



aw.vcx!awFrm as Form 2- methods (...)

Methods described here work in either desktop mode or Web mode: one single call for either mode.

Method	Usage
<code>.wUserLogin()</code> <code>.wUserLogoff()</code>	Logs a user in/out the application; tells FoxInCloud the app. user ID so that it can be injected back when user returns
<code>.wViewParmSet()</code>	Tells FoxInCloud the view parameter name and value, and optionally instructs to query the view
<code>.wSessionVarSet()</code> <code>.wSessionVarGet()</code>	Sets / Gets a variable for the form and the current web user's session that can persist across requests



aw.vcx!awGrd as Grid

Grid is fairly automated in FoxInCloud; you have very few PMs to care about

Property / Method	Usage
<code>.wuVirtualMode</code>	Controls how grid widget loads rows on scroll: <code>.NULL.:</code> auto, <code>.T.:</code> always, <code>>0:</code> mini rows
<code>.wBeforeRowChange ()</code> <code>.wBeforeColChange ()</code>	Move your code from <code>.BeforeRowColChange</code> into these 2 methods to specify action <u>before</u> row OR col change
<code>.wAfterRowChange ()</code> <code>.wAfterColChange ()</code>	Move your code from <code>.AfterRowColChange</code> into these 2 methods to specify action <u>after</u> row OR col change

You can page any grid by dropping it into a `awCntGrdPage` sub-class



aw.vcx!awTxt as TextBox

Textbox extensions let it take advantage of popular Web features

Property / Method	Usage
<code>.wlHTML5placeholder</code>	Set this.ToolTipText as placeholder
<code>.wlDatePicker</code>	If a date is displayed, add a datepicker in Web mode
<code>.wcHTML5type</code>	Specifies a HTML5 type such as <code>color</code> , <code>date</code> , <code>email</code> , <code>month</code> , <code>number</code> , <code>range</code> , <code>search</code> , <code>tel</code> , <code>time</code> , <code>url</code> , <code>week</code>



aw.vcx!awPgf as Pageframe

Pageframe
(AKA 'tabs'
in Web
literature)
gives you
more
control on
Web
behavior

Property / Method	Usage
<code>.wlActivePage*</code>	Control how active page displays in web mode
<code>.wnTab*</code>	Control how page tabs appear and behave in Web mode



aw.vcx!awImg as Image

Image base class is extended to behave almost like a control and include popular Web features

Property / Method	Usage
<code>.DisabledPicture</code>	Emulates standard behavior
<code>.ControlSource</code>	Emulates standard behavior
<code>.wHoverPicture</code>	Picture that displays when hovered by the mouse
<code>.Refresh_()</code>	Makes image participate in <code>.Refresh()</code>