Consultingwerk
software architecture and development

Customizing the OpenEdge Architect Visual Designer

An introduction into the capabilities of the IDesignerHost interface and how it can be accessed from the ABL – or a deep dive into UserControls and inherited Controls

Mike Fechner, Director, Consultingwerk Ltd.

mike.fechner@consultingwerk.de

PUG Challenge Americas

Wednesday, June 8th, 2011

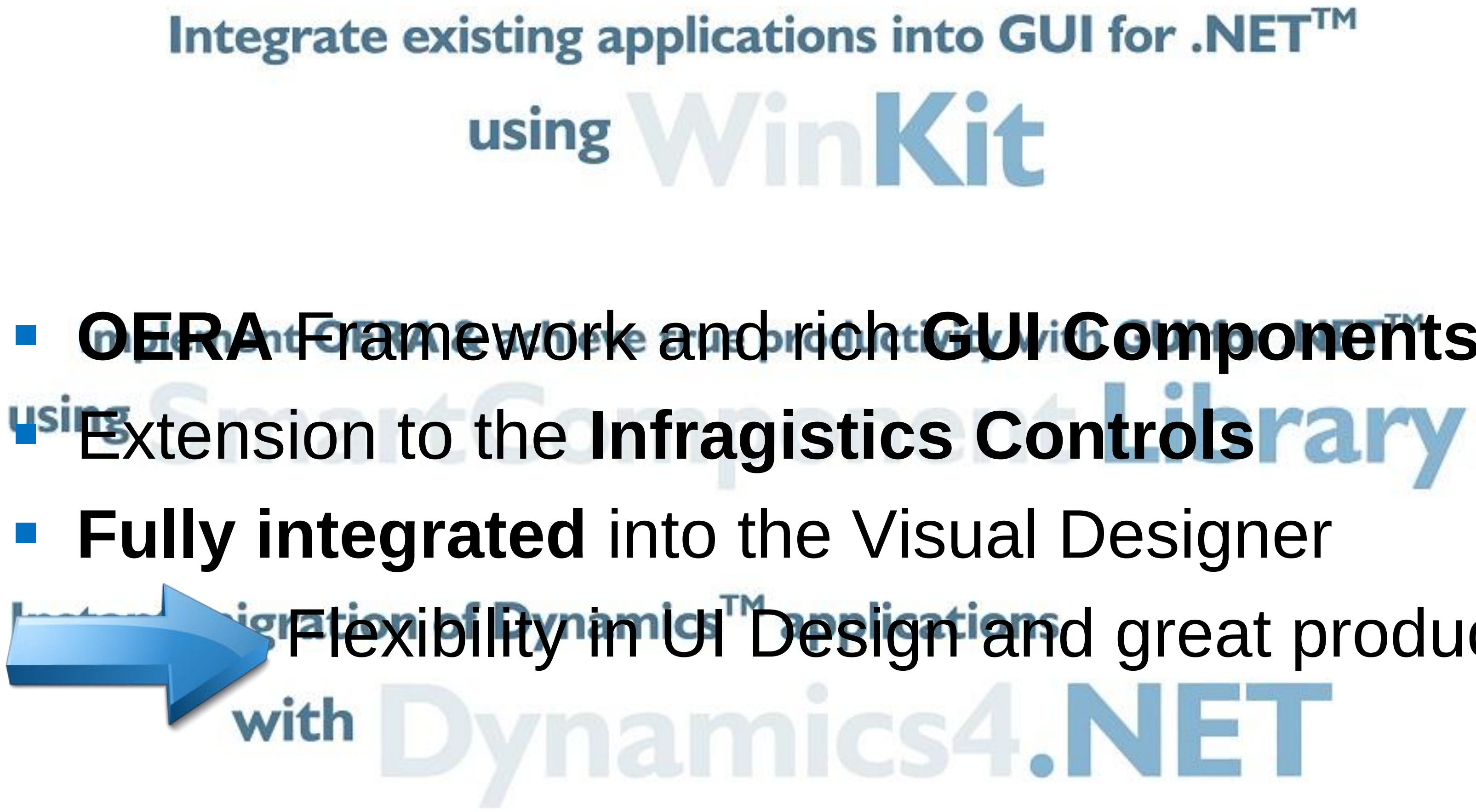
Mike Fechner, Consultingwerk Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **.NET**
- Located in Cologne, Germany
- Vendor of tools and consulting programs
- 21 years of Progress experience (V5 … V10)
- GUI for .NET early adaptor (since 10/2006)

Mike Fechner, Consultingwerk Ltd.

- Customers in Germany, Europe, USA
- Working with small to large Progress Partners and direct end users
- Supporting some of the largest Progress Partners in Germany, Belgium, The Netherlands, Austria and UK with application modernization and user interface technologies
- Network of partnering consultants, like ic4b for Web UI’s, Whitestar Software, DBAppraise

Solutions for the OpenEdge GUI for .NET


Integrate existing applications into GUI for .NET™
using WinKit
using SmartComponent Library
with Dynamics4.NET

- **OERA Framework** and rich **GUI Components**
- Extension to the **Infragistics Controls**
- **Fully integrated** into the Visual Designer

➡ Flexibility in UI Design and great productivity

Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

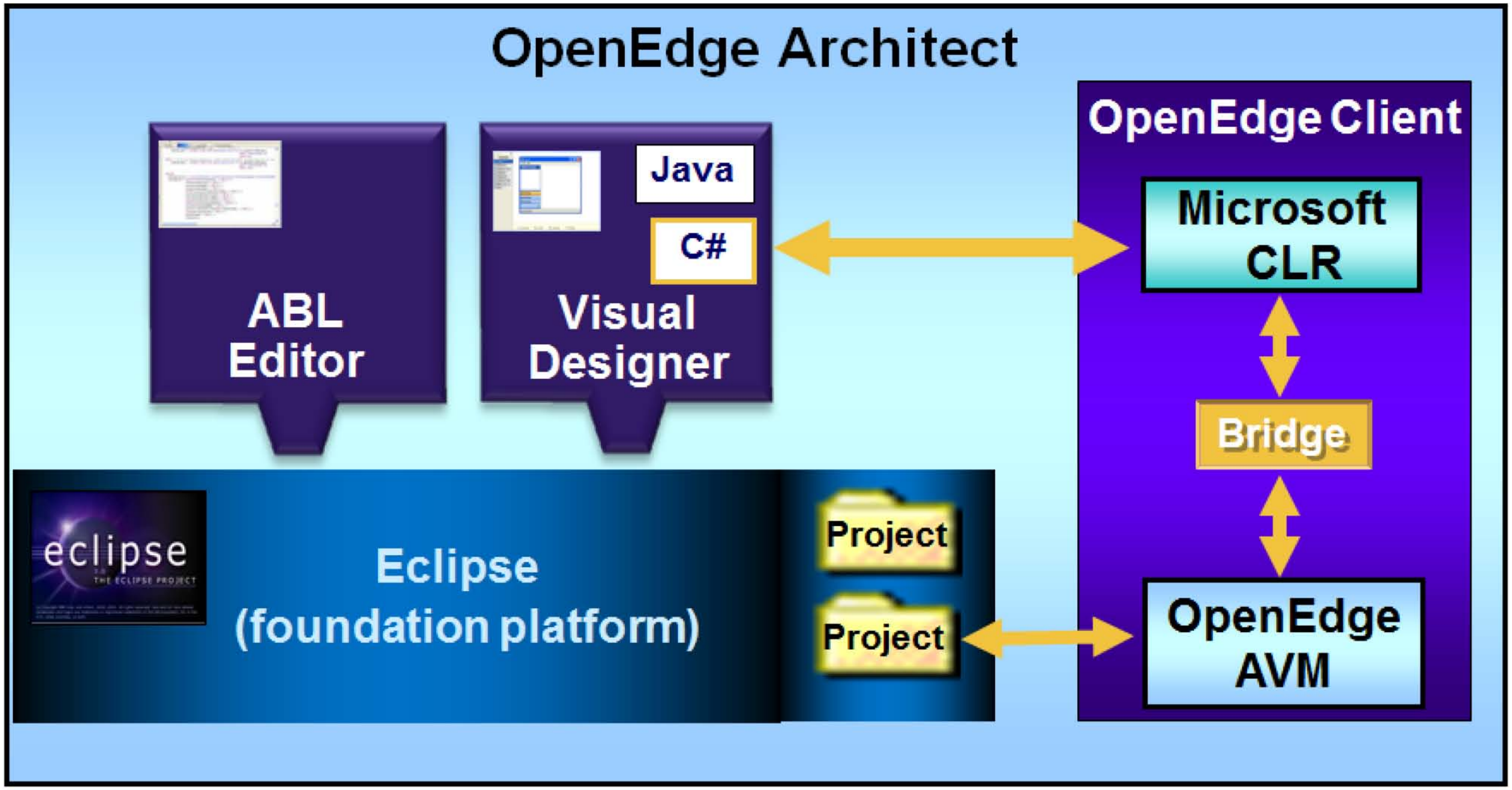
Demo SmartViewerControl Design

- Add customized BindingSource Component to the design canvas
- Use DesignerVerbs to
  - Select Business Entity
  - Select Tables
  - Import Schema
  - Add Fields wizard
- 99% of code is ABL

Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

Visual Designer Architecture



Visual Designer Architecture

- Eclipse Plugin
- Running inside prowin32.exe (project AVM, shared AVM),
- A GUI for .NET application itself
- Based on standard .NET Components for WinForms Designer
  - Design Surface
  - Property Grid
  - …

Visual Designer Architecture

- 3rd Party .NET Controls find all the „services“ they expect from a Visual Designer
  - source for rich design time experience
  - wizards
  - custom property sheets

Visual Designer Architecture

- Root Component (Form, User Control, Inherited Control) is represented by an instance of the base class
- Contained Controls and Components are represented by an instance (running)
- Design time functionality is supported by a Designer instance per Control or Component

Sample

- ABL inherited Control in Visual Designer
  - Message in Constructor
  - Message in Property SETter
  - Raise error from SET validation
  - Review InitializeComponents

Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

IDesignerHost Interface

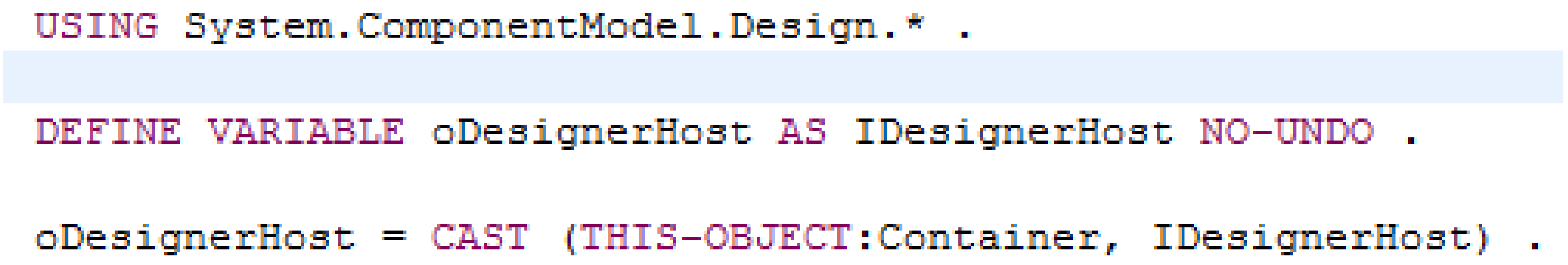
- ***System.ComponentModel.IDesignerHost***
- Microsoft .NET Interface that defines the “glue” in a .NET Visual Designer
- Implemented by PSC in Visual Designer
- Used by 3rd party control vendors to interact with the Design time environment
- ServiceContainer for VD services
- References „RootComponent“
- References „DesignSurface“ Control
- http://msdn.microsoft.com/en-us/library/system.componentmodel.design.idesignerhost.aspx

IDesignerHost Interface

- ***… Visual Designer plays Microsoft rules, not Progress’***
- Core requirement to support „any“ 3rd party Control vendor’s wizards etc. (Infragistics, Telerik, …)

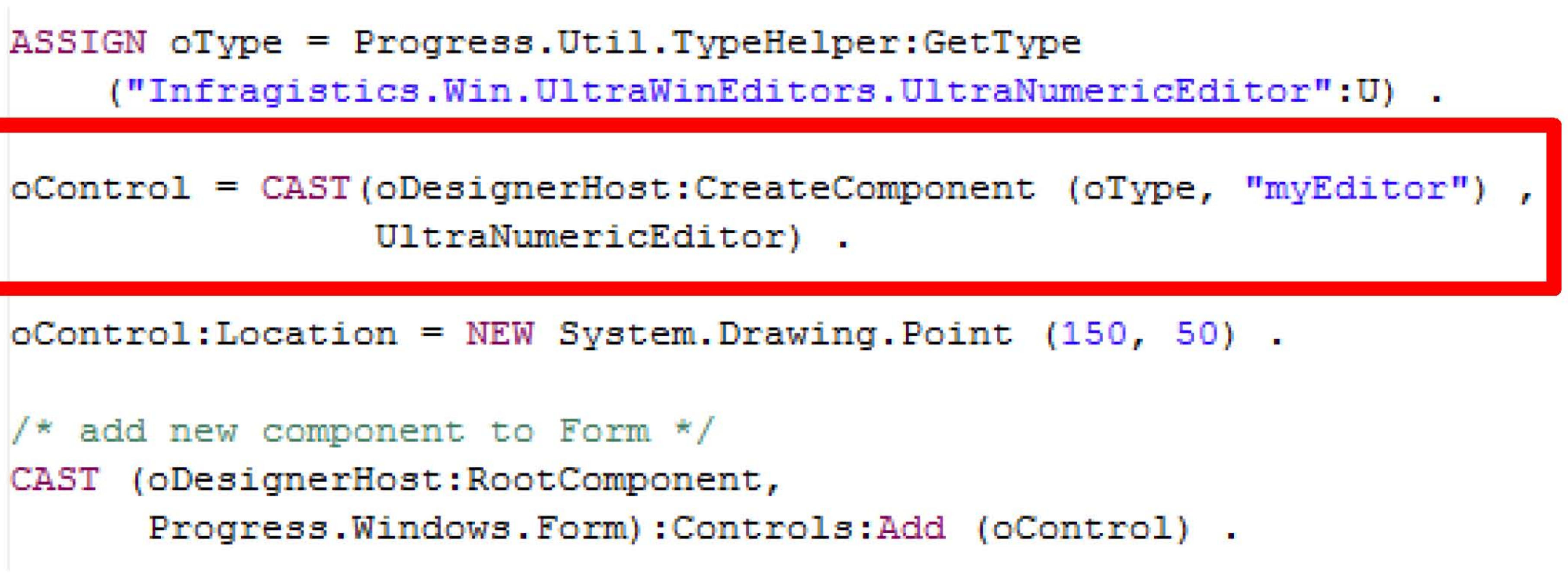
Accessing IDesignerHost Interface

- From a Component:



```
USING System.ComponentModel.Design.* .

DEFINE VARIABLE oDesignerHost AS IDesignerHost NO-UNDO .

oDesignerHost = CAST (THIS-OBJECT:Container, IDesignerHost) .
```

Creating Controls on the Design

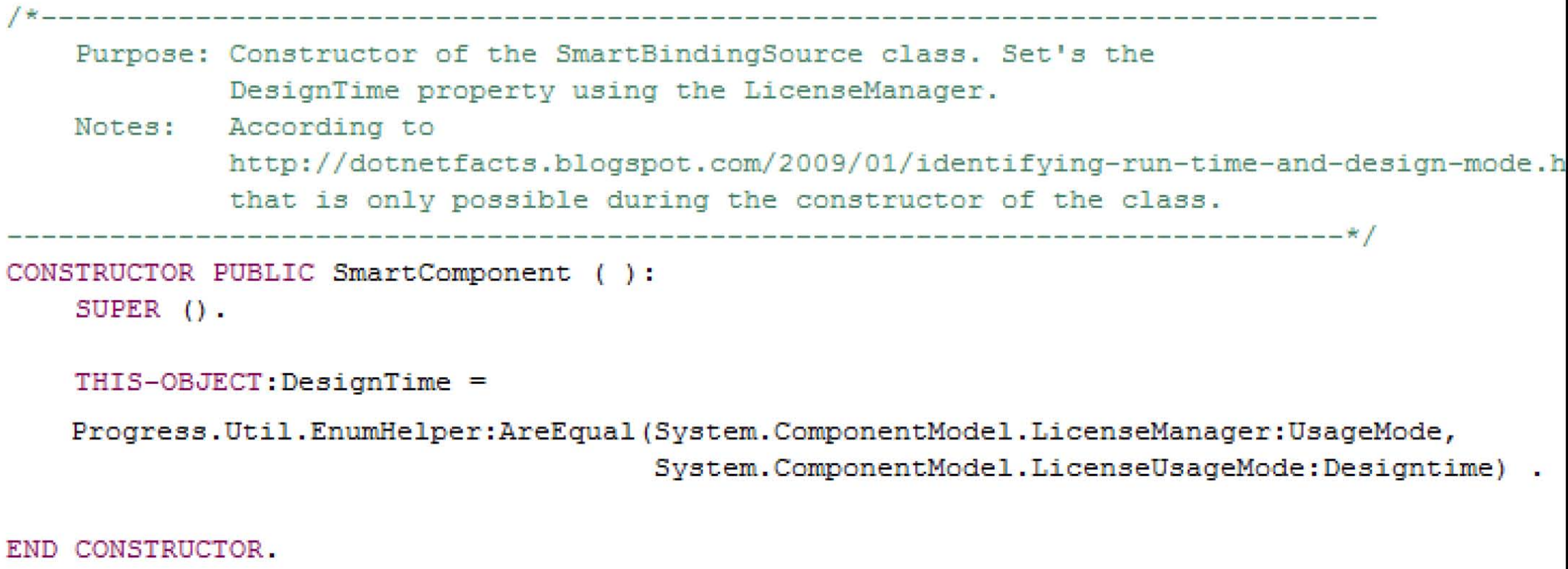


```
ASSIGN oType = Progress.Util.TypeHelper:GetType
    ("Infragistics.Win.UltraWinEditors.UltraNumericEditor":U) .

oControl = CAST(oDesignerHost:CreateComponent (oType, "myEditor") ,
                UltraNumericEditor) .

oControl:Location = NEW System.Drawing.Point (150, 50) .

/* add new component to Form */
CAST (oDesignerHost:RootComponent,
      Progress.Windows.Form):Controls:Add (oControl) .
```

Detecting “Design Time”

- Component: ***DesignMode*** property
  - Does not work for grand childs (Controls in UserControls on Root Component)
- Alternative is checking for ***LicenseManager***‘s context
  - Needs to be done in the constructor
  - Not accessible after the constructor
- http://dotnetfacts.blogspot.com/2009/01/identifying-run-time-and-design-mode.html (thanks to Peter Judge from PSC for sharing that link on PSDN)

Detecting “Design Time”

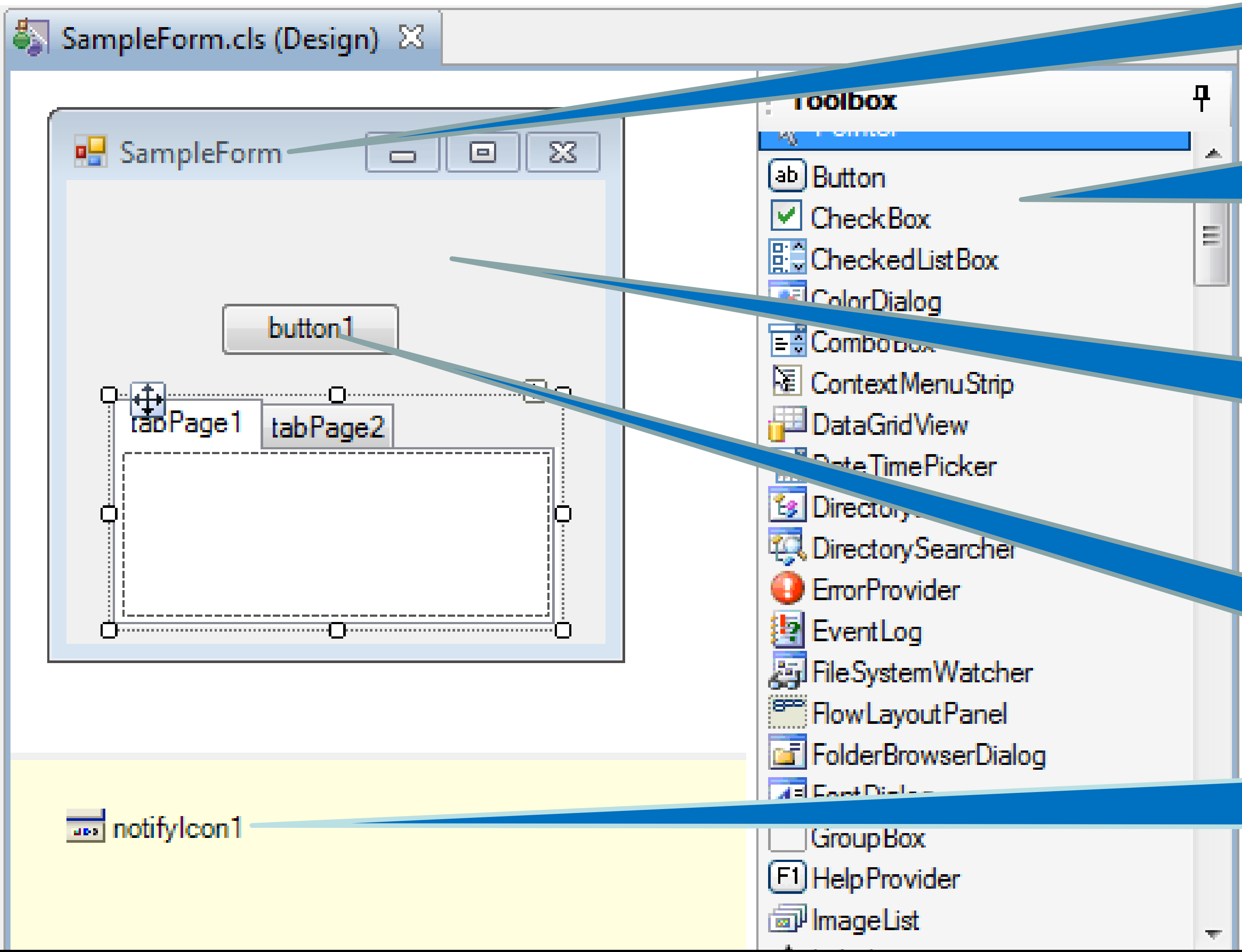


```
/*------------------------------------------------------------------------------
    Purpose: Constructor of the SmartBindingSource class. Set's the
             DesignTime property using the LicenseManager.
    Notes:   According to
             http://dotnetfacts.blogspot.com/2009/01/identifying-run-time-and-design-mode.h
             that is only possible during the constructor of the class.
------------------------------------------------------------------------------*/
CONSTRUCTOR PUBLIC SmartComponent ( ):
    SUPER ().

    THIS-OBJECT:DesignTime =

    Progress.Util.EnumHelper:AreEqual (System.ComponentModel.LicenseManager:UsageMode,
                                       System.ComponentModel.LicenseUsageMode:Designtime) .

END CONSTRUCTOR.
```

Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

Controls and Components



Root Component

Toolbox to drag new Controls on the Form

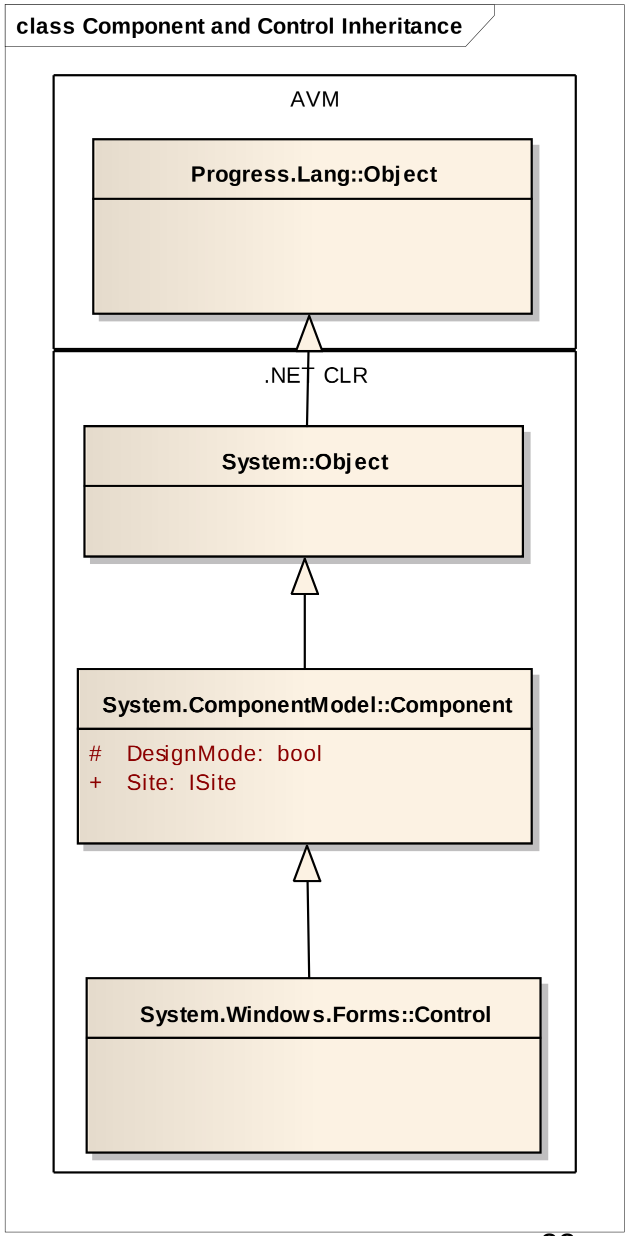
System.ComponentModel.Design**.DesignSurface**

Control on Container

Component on Component area

Controls and Components

- ***Component*** is the base class for all .NET classes that can be used in the Visual Designer
- Component is useful for non visual classes that should be configured from the Visual Design (Controller, Data Access, etc.)
- ***Controls are Components*** that can be place on top of the root control (Form or UserControl)


class Component and Control Inheritance
AVM
Progress.Lang::Object
.NET CLR
System::Object
System.ComponentModel::Component
# DesignMode: bool
+ Site: ISite
System.Windows.Forms::Control

Controls and Components

- Controls need to have constructor with NO parameter (default constructor)
  - IDesignerHost:CreateComponent does not use constructor parameters
  - Generated code does not use parameters for constructor
- ABL Component needs to have constructor with IContainer parameter and Default constructor
  - Default constructor used by Visual Designer
  - IContainer construtor used by generated source code (10.2B, not in 10.2A)

ABL Components and Controls

- Root Component:
  - ABL inherited Form
  - ABL inherited UserControl
  - ABL inherited Control
  - Root component is represented by an instance of the base class
- Contained Components
  - ABL UserControl
  - ABL inherited Control
  - ABL inherited Component

Visibility of ABL Components/Controls

- Inherited properties and events
- ABL properties of basic (primitive) datatypes: Character, Integer, Logical, Date, …
- ABL properties of .NET Types
  - classes
  - interfaces
- ABL events **only** when based on .NET delegate
  - System.EventHandler derived
  - ABL events based on signature are not supported
  - Makes bad practice good ☹ (unable to specialize events)

Visibility of ABL Components/Controls

- Translation of INITIAL Option on the PROPERTY Definition to the DefaultValue Attribute
- Properties with Value = DefaultValue won‘t be written to InitializeComponents
- Indicated by **bold** font in the Property Sheet
- Try to avoid dynamic INITIAL value in Constructor of the Component, the Visual Designer would always write it to the source code (no longer dynamic anymore)

Visibility of ABL types

- ABL (hybrid) class represented by .NET System.Type object
  - only once it has been newed
  - Important to know, when passing System.Type of ABL Control to IDesignerHost:CreateComponent!
- ABL properties, methods and events not visible to .NET side at runtime
- Exception: If they are part of a .NET Interface a class is implementing
  - Helps when implementing DataBinding of custom properties
  - Only chance to expose ABL property to .NET

Demo

- Root Component functionality:
  - SmartWindowForm
  - Alternative Dialog for adding custom controls to a Form
- Property Sheet of ABL inherited Control, ability to „connect“ two instances
  - using .NET Interfaces implemented by the Controls

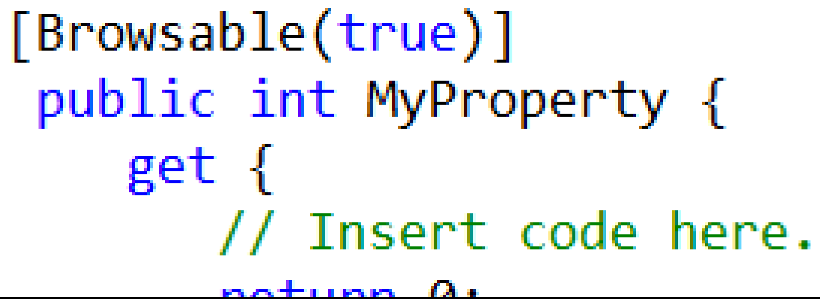
Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

Type and Property Attributes

- A ***Type*** is a synonym for a .NET Class
- Classes consist of (among others)
  - Methods, Properties (DEFINE PROPERTY), Events
- Types, ***Properties*** and ***Events*** have ***Attributes*** that define how they should be handled in the Visual Designer (or other cool .NET features ☺, such as Serialization)
- Annotations in C# source code

Property Attribute as Annotation in C#



```
[Browsable(true)]
 public int MyProperty {
     get {
         // Insert code here.
         return 0;
```

Type Attributes, samples

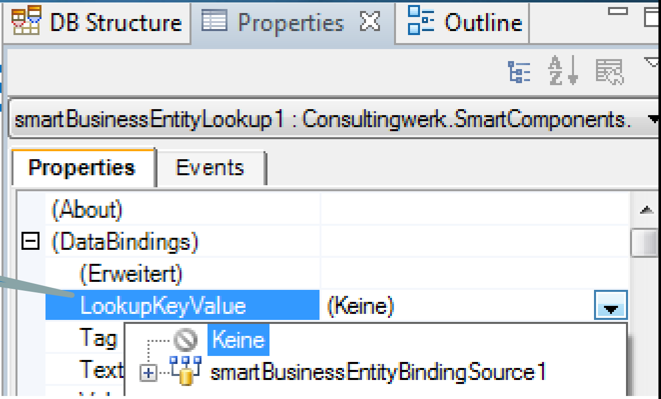
- DefaultEventAttribute
  Default Event used when double clicking on a Control in the Visual Designer
- DefaultPropertyAttribute
  Default Property selected in the Property Grid
- DesignerAttribute:
  Name of the type ComponentDesigner for this class (loosely typed)
- http://msdn.microsoft.com/en-us/library/a19191fh.aspx, http://msdn.microsoft.com/en-us/library/tk67c2t8.aspx, http://msdn.microsoft.com/en-us/library/ms171724.aspx

Consultingwerk
software architecture and development

Property Attributes, samples

- BindableAttribute
  Appears in Data Bindings
- BrowsableAttribute
  Visible/Hidden in Property Grid
- EditorAttribute
  Type of Editor in Property Grid
- CategoryAttribute
- DefaultValueAttribute
- DescriptionAttribute

Bindable Property (ABL)



DB Structure | Properties | Outline

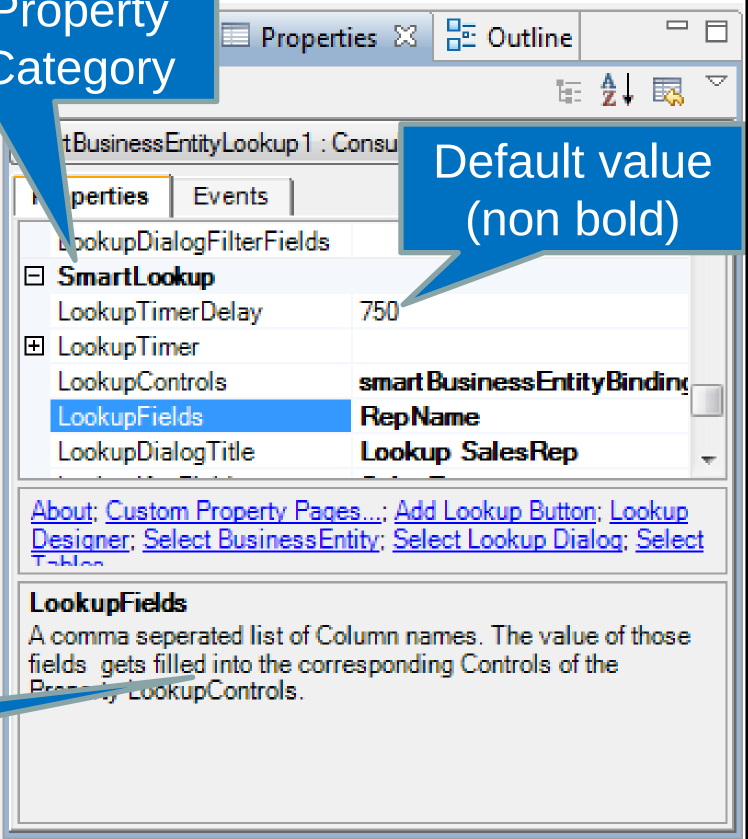
smartBusinessEntityLookup1 : Consultingwerk.SmartComponents.

Properties | Events

(About)	
⊟ (DataBindings)	
(Erweitert)	
LookupKeyValue	(Keine)
Tag	Keine
Text	smartBusinessEntityBindingSource1

Property Category

Default value (non bold)



Properties | Outline

tBusinessEntityLookup1 : Consu

perties | Events

LookupDialogFilterFields	
⊟ SmartLookup	
LookupTimerDelay	750
⊞ LookupTimer	
LookupControls	**smartBusinessEntityBinding**
LookupFields	**RepName**
LookupDialogTitle	**Lookup SalesRep**

About; Custom Property Pages...; Add Lookup Button; Lookup Designer; Select BusinessEntity; Select Lookup Dialog; Select Tables

LookupFields
A comma seperated list of Column names. The value of those fields gets filled into the corresponding Controls of the Property LookupControls.

Property Description

Setting these Attributes from ABL?

- ABL does not support the annotation syntax ☹
- Everybody, please remind PSC that they should!!!
- ABL exposes just primitive type properties (basic data types, no references to ABL objects), or references to .NET objects (.NET class / interface reference), ABL controls may implement .NET interface
- All public properties Browsable (no way to hide runtime only properties)

Setting these Attributes from ABL?

- ***System.ComponentModel.TypeDescriptor*** static class provides standard view to the type and property attributes
  - GetEvents (Object object)
  - GetEvents (Type type)
  - GetProperties (Object object)
  - **GetProperties (Type type)**
  - …

Setting these Attributes from ABL?

- ***System.ComponentModel.ICustomTypeDescriptor***
  - Interface allows a class (instance) to return customized property attributes at runtime ☺
  - may use TypeDescriptor as a source of information
  - Consider caching to avoid negative performance, consider using .NET code to put together information
- http://msdn.microsoft.com/en-us/library/system.componentmodel.icustomtypedescriptor.aspx
- All your ABL custom controls and components may need to implement this, consider using Include files to make up the lack of multiple inheritance (**ICustomTypeDescriptor.i**)
- Use PROTECTED properties (i.e. NonBrowsableProperties, PropertyCategories) to store information as an alternative to annotations

Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

Hiding Properties from the Property Grid

- Use case: Runtime only properties (status, handles, etc.)
- May avoid errors from Form‘s ***InitializeComponents*** method when developer did set unexpected values
- Avoid developer confusion: Focus only on design time relevant stuff
- Requires setting ***Browsable*** attribute to FALSE

Sample

- ICustomTypeDescriptor implementation in the ABL

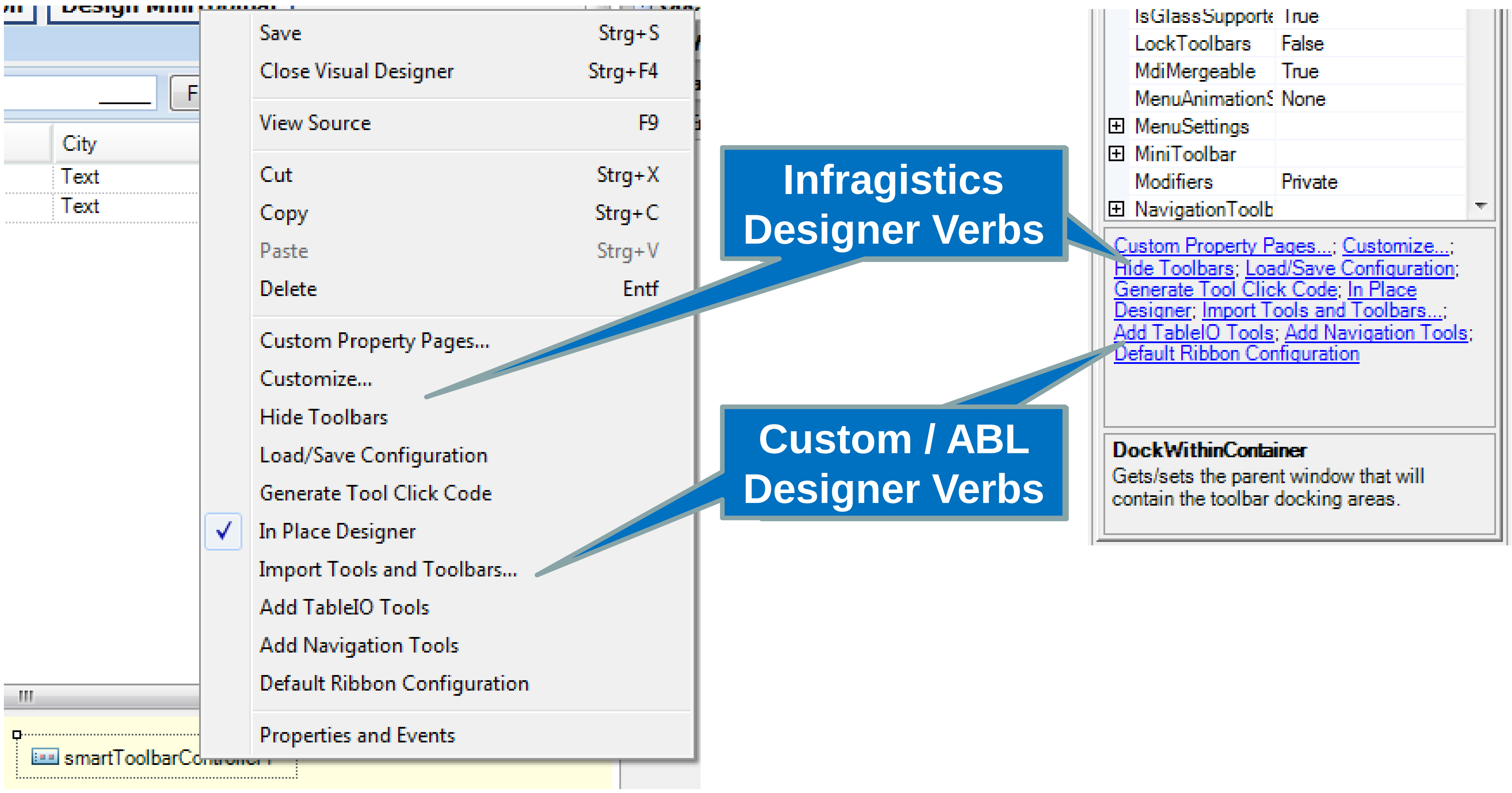
Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

Designer Verbs

- A ***Designer Verb*** is an activity a developer could perform with a (design time) instance of a Component or the Component Designer
- Designer Verbs allows to invoke wizard like functionality or custom dialogs inside the Visual Designer
- Accessible from hyper-links in the property grid or the context menu or the design time instance

Designer Verbs


Save Strg+S
Close Visual Designer Strg+F4
View Source F9
Cut Strg+X
Copy Strg+C
Paste Strg+V
Delete Entf
Custom Property Pages...
Customize...
Hide Toolbars
Load/Save Configuration
Generate Tool Click Code
✓ In Place Designer
Import Tools and Toolbars...
Add TableIO Tools
Add Navigation Tools
Default Ribbon Configuration
Properties and Events
City
Text
Text
smartToolbarController1
Infragistics Designer Verbs
Custom / ABL Designer Verbs
Custom Property Pages...; Customize...; Hide Toolbars; Load/Save Configuration; Generate Tool Click Code; In Place Designer; Import Tools and Toolbars...; Add TableIO Tools; Add Navigation Tools; Default Ribbon Configuration

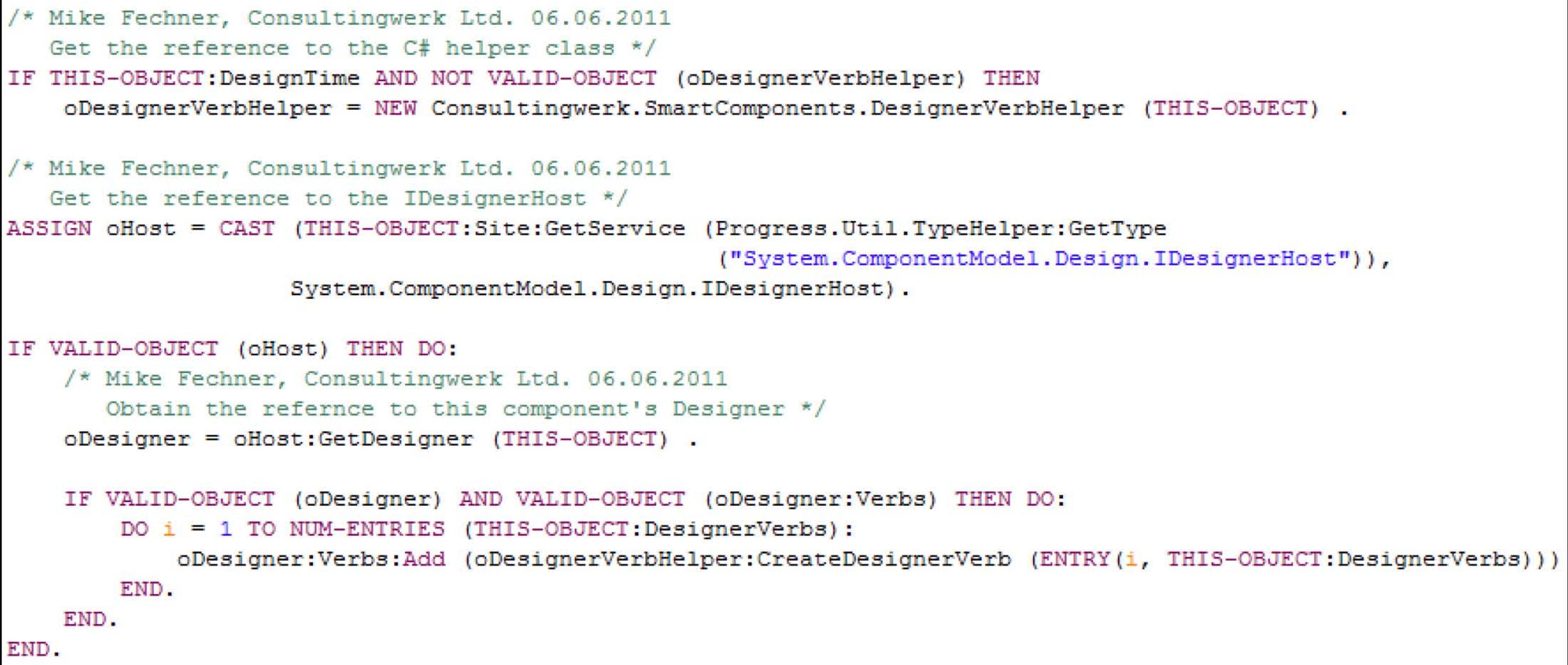
IsGlassSupporte	True
LockToolbars	False
MdiMergeable	True
MenuAnimationS	None
⊞ MenuSettings	
⊞ MiniToolbar	
Modifiers	Private
⊞ NavigationToolb	

DockWithinContainer
Gets/sets the parent window that will contain the toolbar docking areas.

Designer Verbs

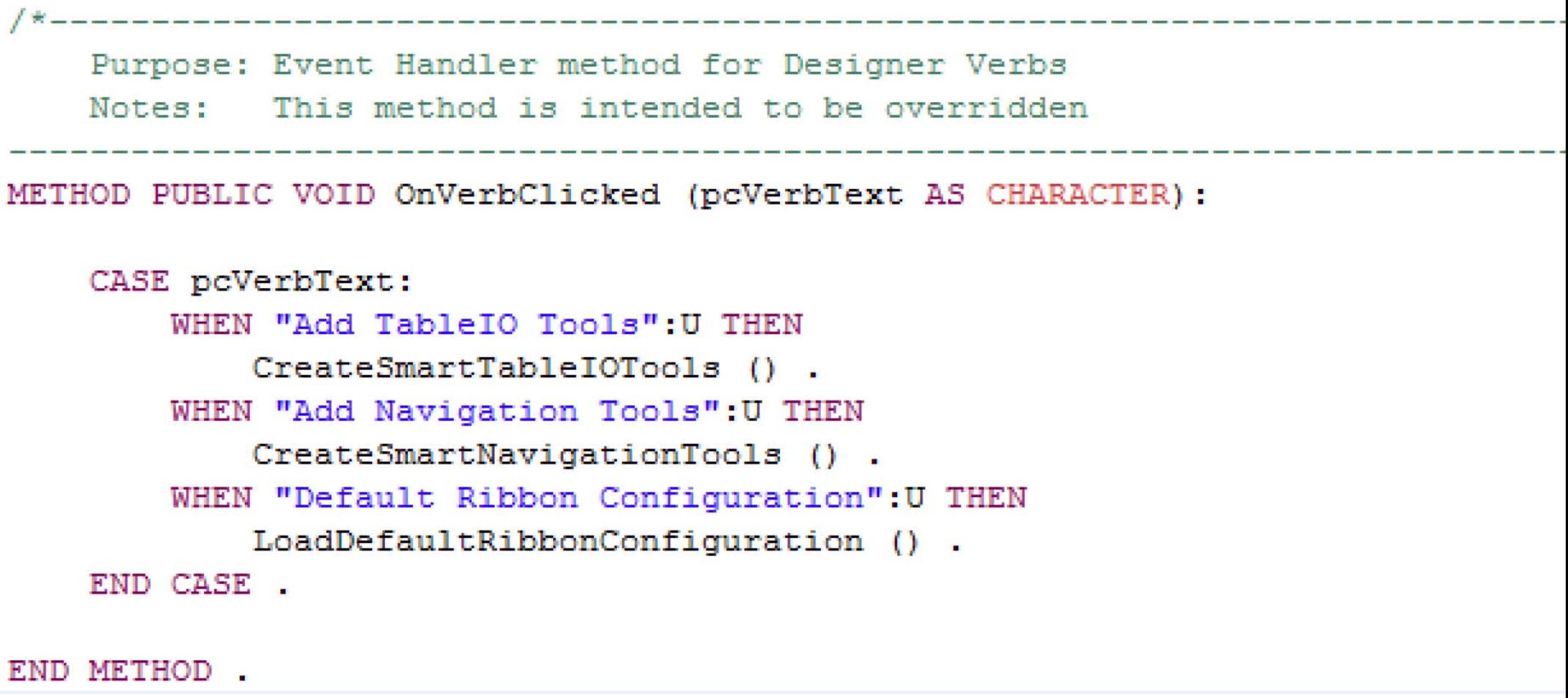
- Public Collection of DesignerVerbs on the Component Designer
- ***System.ComponentModel.Design.DesignerVerb*** class
- Designer Verb constructor requires a „Text“ for the label and a System.EventHandler (delegate = function pointer)
- Unfortunately this is one of the few limitations of the ABL GUI for .NET bridge: We cannot implement delegates ☹
- Solution: C# helper class that creates the Verb and signals the Component using an Event or callback into ABL code (requires .NET Interface)
  – acceptable workaround

Designer Verbs



```
/* Mike Fechner, Consultingwerk Ltd. 06.06.2011
   Get the reference to the C# helper class */
IF THIS-OBJECT:DesignTime AND NOT VALID-OBJECT (oDesignerVerbHelper) THEN
    oDesignerVerbHelper = NEW Consultingwerk.SmartComponents.DesignerVerbHelper (THIS-OBJECT) .

/* Mike Fechner, Consultingwerk Ltd. 06.06.2011
   Get the reference to the IDesignerHost */
ASSIGN oHost = CAST (THIS-OBJECT:Site:GetService (Progress.Util.TypeHelper:GetType
                                                  ("System.ComponentModel.Design.IDesignerHost")),
                    System.ComponentModel.Design.IDesignerHost).

IF VALID-OBJECT (oHost) THEN DO:
    /* Mike Fechner, Consultingwerk Ltd. 06.06.2011
       Obtain the refernce to this component's Designer */
    oDesigner = oHost:GetDesigner (THIS-OBJECT) .

    IF VALID-OBJECT (oDesigner) AND VALID-OBJECT (oDesigner:Verbs) THEN DO:
        DO i = 1 TO NUM-ENTRIES (THIS-OBJECT:DesignerVerbs):
            oDesigner:Verbs:Add (oDesignerVerbHelper:CreateDesignerVerb (ENTRY(i, THIS-OBJECT:DesignerVerbs)))
        END.
    END.
END.
```

Designer Verbs



```
/*-----------------------------------------------------------------------
    Purpose: Event Handler method for Designer Verbs
    Notes:   This method is intended to be overridden
-------------------------------------------------------------------------
METHOD PUBLIC VOID OnVerbClicked (pcVerbText AS CHARACTER):

    CASE pcVerbText:
        WHEN "Add TableIO Tools":U THEN
            CreateSmartTableIOTools () .
        WHEN "Add Navigation Tools":U THEN
            CreateSmartNavigationTools () .
        WHEN "Default Ribbon Configuration":U THEN
            LoadDefaultRibbonConfiguration () .
    END CASE .

END METHOD .
```

-IOEverywhere 1

- Use whenever you can ☺
- Lifts restriction of not usable WAIT-FOR statement etc. in functions and non-void methods
- Required for showing ABL Dialogs from Designer Verbs (we had to use .NET Dialogs before)
- Undocumented startup parameter since 10.2B02
- … probably documented and new default in OE11
- Use it on the project AVM/shared AVM for use in the Visual Designer

Consultingwerk
software architecture and development

Agenda

- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

ABL Data-Bindable Properties

- Code review
  - Setting Bindable(True) attribute
  - Implementing .NET Interface with that property

Agenda

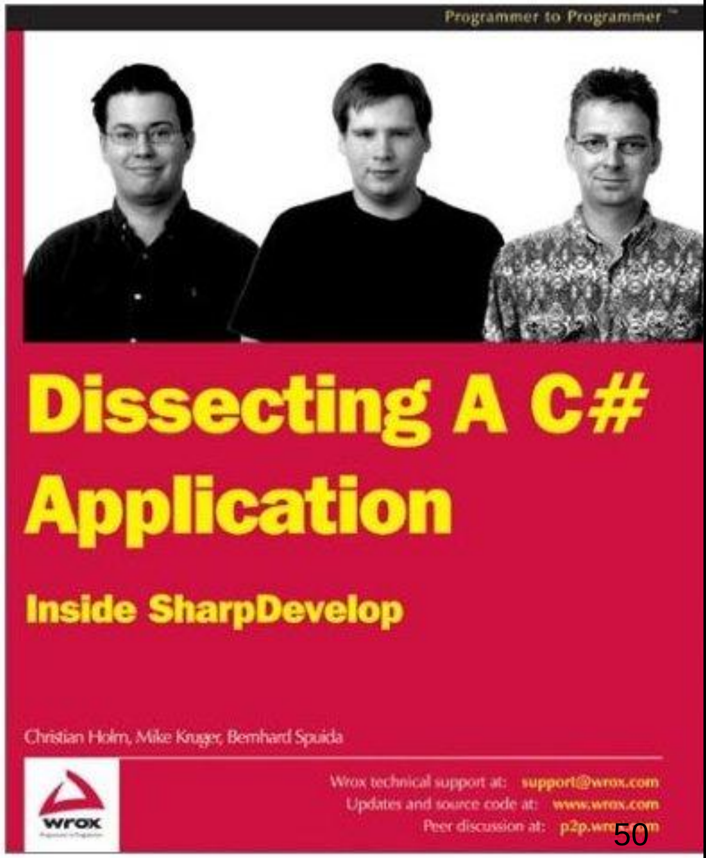
- Demo SmartComponent Library Viewer Design
- Visual Designer Architecture
- IDesignerHost Interface
- Controls and Components
- Type and Property Attributes
- Hiding Properties from the Property Sheet
- Designer Verbs
- Demo: ABL Data-Bindable Properties
- Demo: ABL based Visual Designer

ABL based Visual Designer

- Hosted in custom application
- Code review:
  - putting it all together

More information

- Google, Codeproject, MSDN, ***PSDN***, …
- I didn‘t find a book on MSDN press
- However there‘s a book from the team that wrote SharpDevelop - an open source alternative to MS Visual Studio: “Dissecting a C# Application: Inside SharpDevelop”
  - Has a chapter on building a WinForms Designer


Programmer to Programmer
Dissecting A C# Application
Inside SharpDevelop
Christian Holm, Mike Kruger, Bernhard Spuida
Wrox technical support at: support@wrox.com
Updates and source code at: www.wrox.com
Peer discussion at: p2p.wrox.com
wrox

- ***Used by Progress to build the Visual Designer …***

More links…

- Posted by Matt Baker from PSC in a recent PSDN discussion
  - The perfect host: Create And Host Custom Designers With The .NET Framework 2.0 http://msdn.microsoft.com/en-us/magazine/cc163634.aspx
  - .NET Shape Library: A Sample Designer http://windowsclient.net/articles/shapedesigner.aspx
  - Designer Serialization Overview http://msdn.microsoft.com/en-us/library/ms171834.aspx

Questions