IBUC 2023
DURHAM, NC, USA

Blaise 5
Data Entry Verification & ZoneDef samples

Background (1)

- There are certain core items in the Blaise 5 suite of programs
  - Windows & browser DEPs
  - iOS & Android apps
  - Control Centre
  - Data & Meta Viewers
  - Converters

- Other items are provided as samples (with the source code)
  - SAS/SPSS/Stata export
  - CMA
  - Data Entry Verification
  - Zone definer
  - Other API demo apps

Background (2)

- Samples are designed to
  - show the user how to achieve certain things
  - allow the user to adapt the sample to fit their own use-case
- History
  - There was a period where certain items were deemed to be something more than samples but less than core
  - This idea has been reversed and samples should now ship with the source code
- Patching
  - Allowing the user to update the source code also allows the user to produce a patch which can be shared with other users, e.g.
    - someone translates the user interface in the Data Verification Tool
    - someone adds a new export function to the SAS/SPSS/Stata sample

What is Data Entry Verification?

- Also called Double Data Entry
- First data entered by operator; second pass by verifier
- Verifier has 2 options for verification:
  - Blind: first pass data not visible; if change is made select previous or current value
  - Open: data is visible; if change is made then verifier can select a reason for the changed value
- Uses separate *verification* datamodel
  - Content/Look of verification dialog
  - Content of log file

Data Entry Verification (1)

- This program provides facilities similar, but not identical, to what can be found in Blaise 4
  - it's not identical because of differences in the architecture of the two versions
    - e.g. Ditto is not available as a base-action

- It is set up as a custom Windows DEP program
  - it uses the API (therefore requiring an API license)
  - the user interface is written in WPF
  - it shares a DLL with the Zone definer program to provide various shared items
    - e.g. the records browser and the menu

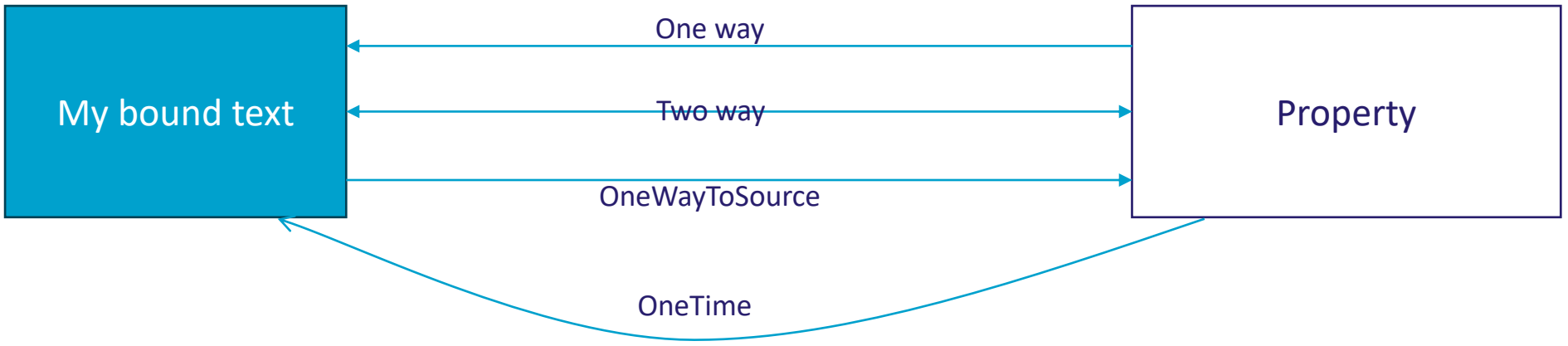
Data Entry Verification (2)

- History
  - initial setup created pre-corona time
  - various new items were needed in the infra
  - no-one seemed to use it

- Recent activity
  - someone complained it didn’t work
  - after reviewing the code and how it had been delivered, a rewrite took place
  - this rewrite tightened up the code-base somewhat, allowed using a remote server as the source instead of just a local one and converted the coding pattern to MVVM

Data binding

- WPF (amongst others) allows controls to be linked to a variable in the codebase
- When the value of the variable changes, it can be reflected in the display
- When an event occurs in the display, it can be reflected back to the codebase

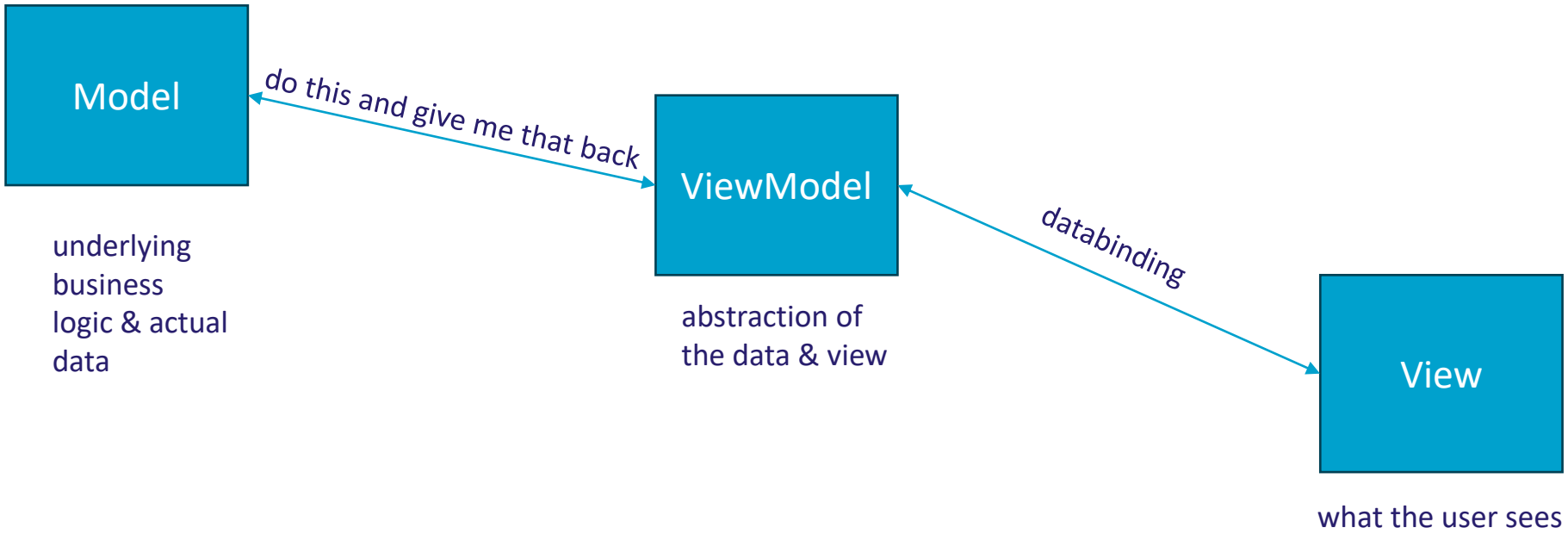


Model-View-ViewModel (1)

- MVVM is yet another programming architectural model, c.f.
  - MVC (model-view-controller)
  - MVP (model-view-presenter)
  - MVB (model-view-binder)
- Came out of Microsoft ca. 2005
- Leverages XAML and databindings

Model-View-ViewModel (2)

- There are many websites that you can find that will define what MVVM is, how it works and what it's advantages are
- Some of these sites are actually useful but to reduce it to a basic explanation



Model-View-ViewModel (3)

- Infra
  - you bind to a property
  - to get notifications about updates you implement INotifyPropertyChanged in the ViewModel

```
public event PropertyChangedEventHandler PropertyChanged;
protected virtual void OnPropertyChanged(string propertyName)
{
  if (PropertyChanged != null)
  {
    PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
  }
}
```

Model-View-ViewModel (4)

- Set up property to implement OnPropertyChanged

```
private Visibility _optionsVisibility = Visibility.Collapsed;

public Visibility OptionsVisibility
{
  get { return _optionsVisibility; }
  set {
    if (_optionsVisibility == value)
      return;
    _optionsVisibility = value;
    OnPropertyChanged("OptionsVisibility");
  }
}
```

Model-View-ViewModel (5)

- Bind in the XAML

```
BOptionsVisibility="{Binding DataContext.OptionsVisibility,
                        RelativeSource={RelativeSource AncestorType={x:Type Window}}}"
```

- BOptionsVisibility is a DependencyProperty in the user control that implements the menu of buttons
- Here it is bound to the OptionsVisibility property in the data context
- In the user control, BOptionsVisibility is bound to the Visibility of the image on the button itself

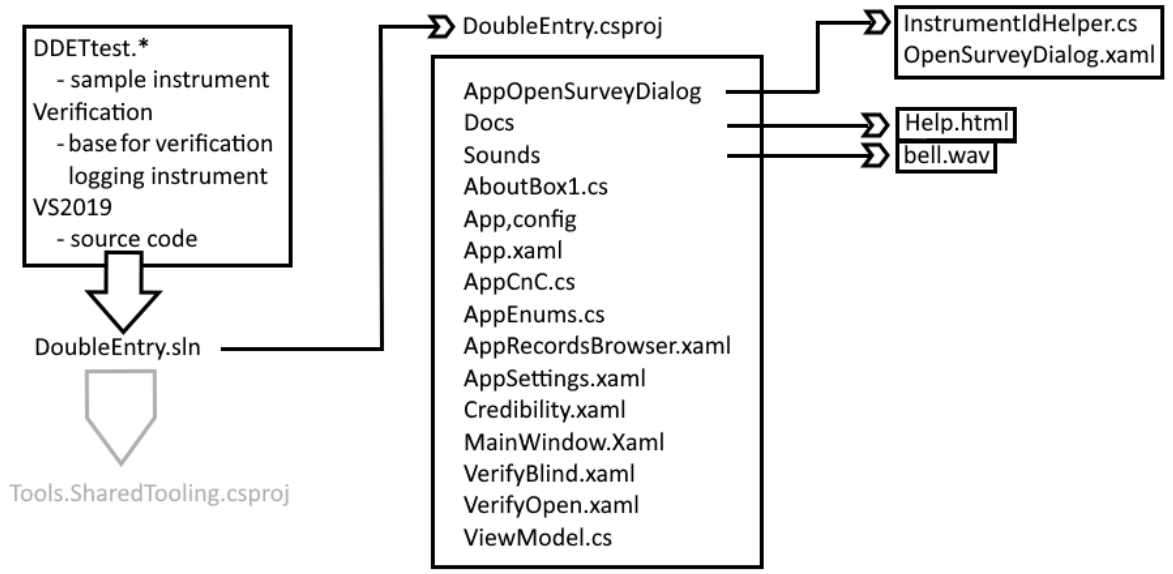
```
Visibility="{Binding BOptionsVisibility,
            RelativeSource={RelativeSource AncestorType=UserControl}}"
```

Model-View-ViewModel (6)

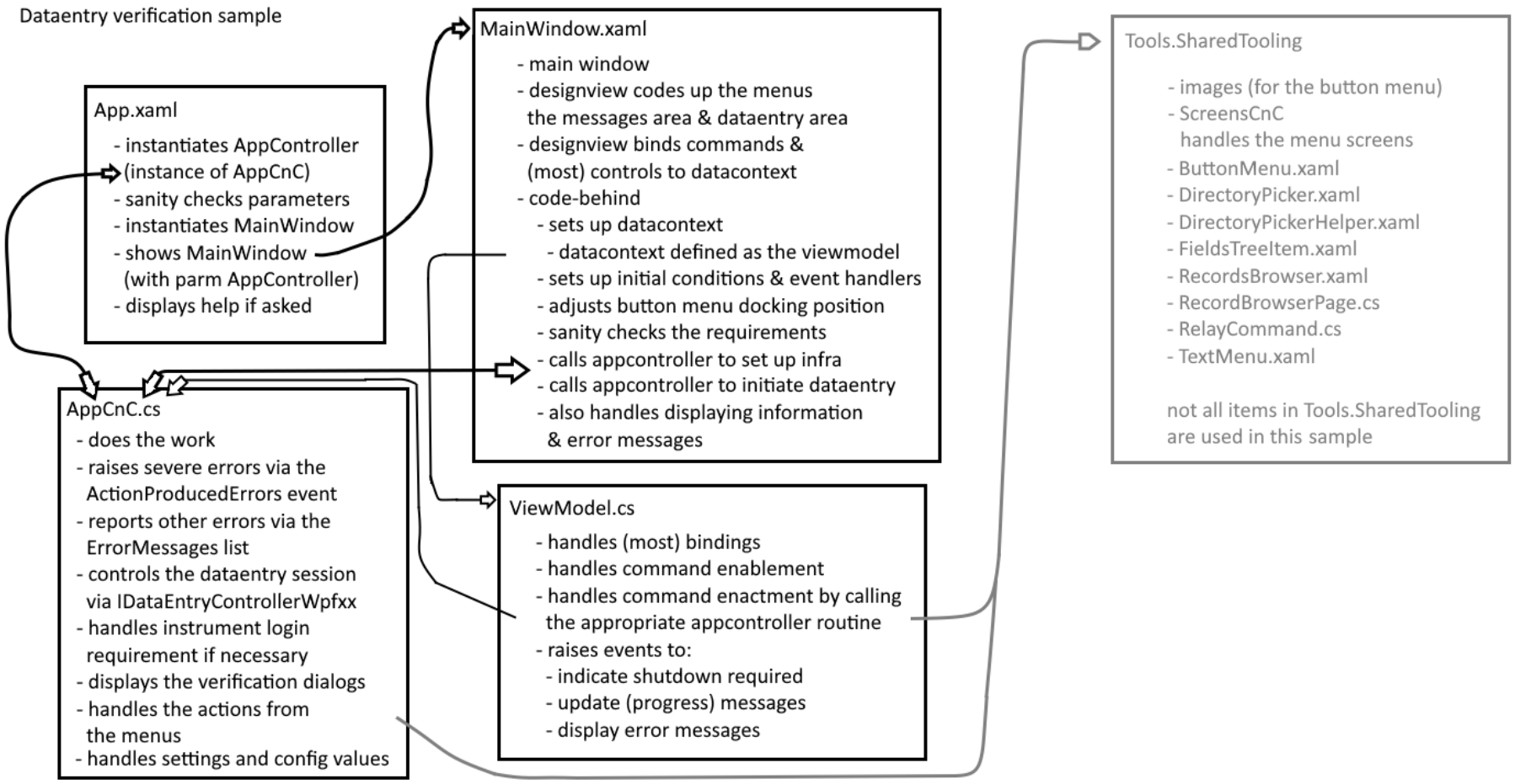
- DataContext property of .NET FrameWork:
  - defines where to find the bound variables
- DataContext = this;
  - sets the datacontext to the current module
  - properties to be defined in the current module
- DataContext = <instance-of-viewmodel-object>
  - sets the datacontext to the specified instance
  - properties to be defined in the specified instance

Data Entry Verification (1)

Dataentry verification sample



Data Entry Verification (2)



Data Entry Verification (3)

- There is a minimal set of requirements that must be satisfied before anything will happen
  - server, appropriately authorized id and password, port number, protocol
  - instrument, bdix (to point to the actual data)
  - location of the verification datamodel (a default setup is supplied with Blaise 5)
  - location of the log file
  - a special dataentrysettings entry if *blind* verification is used: *Existing forms but create an empty*

- Extras
  - there is a minimal records browser
    - searching works on the page, not on the whole dataset, and on the first field of the primary key
    - ordering works on the page, not on the whole dataset
    - it has two modes – show everything at once/show in pages
      - if you have a lot of records then showing everything at once will break your memory – there is no check on this
  - positioning of menu

Data Entry Verification (4)

Demo

Data Entry Verification (5)

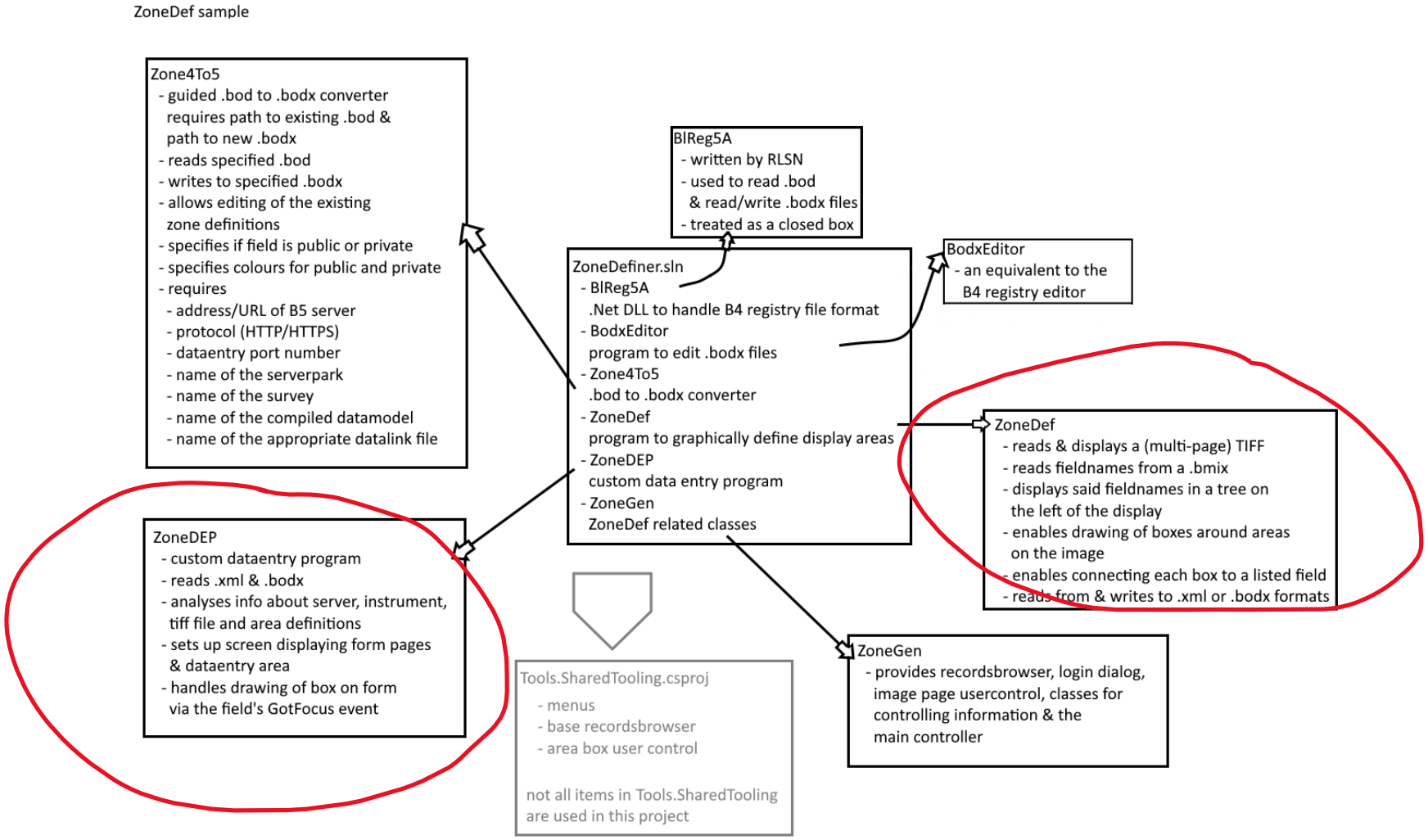
- If this doesn't do what you want, you have the source code – adapt it
- If you find what you think is a bug – fix it and let us know you have done and how
- If you think your adaptation will be useful for other people – make a patch and put it in the Blaise store

- Think about the session data – if you don't throw it away at the end of the data verification session, you'll get a mismatch in the pop-ups
  - because on startup, the record is read from the real database, not the session database

Zone definer (1)

- In Blaise 4 there is a feature called Zonedef which allows you to display an active image (multipage-TIFF) alongside a data entry session where corresponding areas of the image are highlighted when a field is entered
- This feature relies on a third-party control
- In Blaise 5, as with the Data Entry Verification program, this feature is implemented as a custom DEP sample and utilizes the API
- The utility uses C# and Blaise 5
  - the image control is based on native C# controls
  - highlighting is based on native C# controls
  - zone definition can be achieved by a visual editor or, if a migrated B4 definition file is used, via a program akin to the B4 Registry Editor

Zone definer (2)

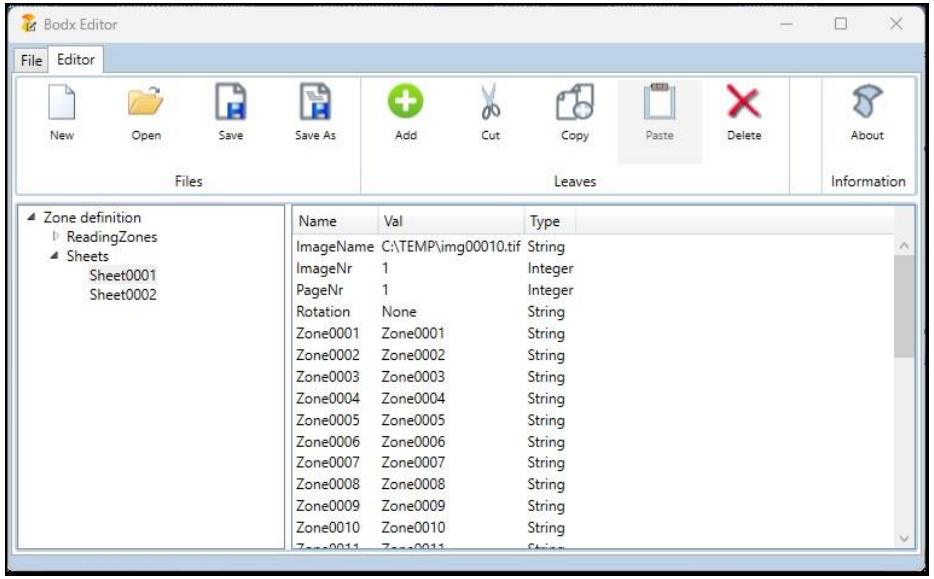


Zone definer (3)

- *.bod to .bodx – Zone4To5, A simple conversion wizard*
- In B4, ZoneDef settings are held in a specific type of file - .bod; in B5 it is .bodx or .xml
- The B5 variant has extra items to hold things like the servername and port number but also the new privacy flag
- Zone4To5 shows various panels to display various items needed for conversion; it does very few clever things with WPF and is an easy/medium program
- You can do the Exercises that are listed in the comments

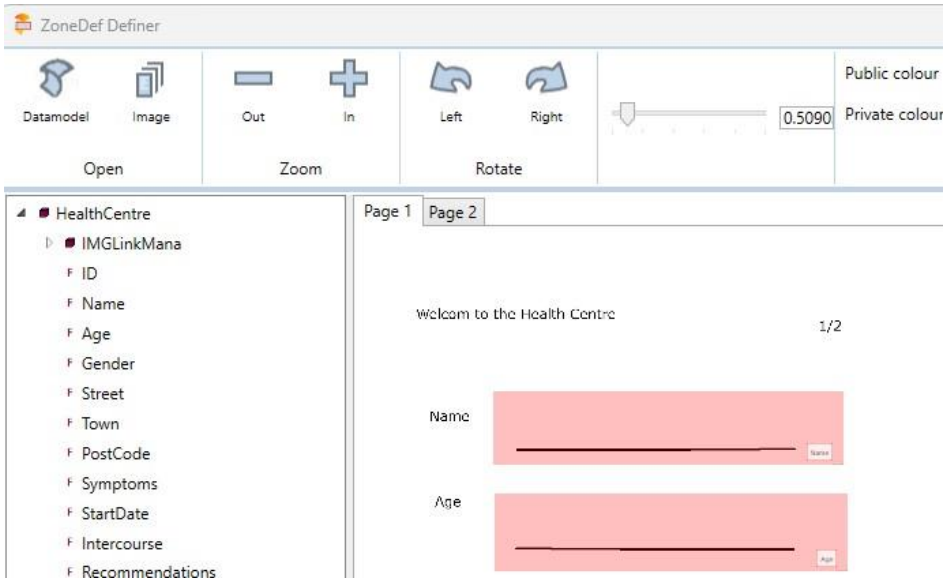
Zone definer (4)

- *.bodx Editor - A simple editor for a .bodx file*
- Implements New (item), Open file, Save file, Save file As, item cut, copy, paste, delete
- View is conceptualized as a tree with leaves - ObservableCollection of a class, Leaf
- Leaf holds current level Properties and subleaves
- Undo – uses a stack of ObservableCollection of a class, Leaf; each change saves the collection which means undo is merely a case of popping the stack and refreshing the appropriate bindings
- Note that *redo* is left as an exercise for the reader (advanced programming trail – see later)



Zone definer (5)

- *Visual zone definer – lets you define highlighting zones via drag & resize*
- IMGMana block has to be defined by hand (if used)
- Save in .bodx or .xml format
- Privacy setting to hide parts of the image
  - note: this does NOT hide the data in the instrument
- Click on the image and drag out an area
  - left-click is public, right-click is private
  - click and hold to drag; resize from bottom right corner
- Click on a field to associate it with an area
- You can do the Exercises that are listed in the comments



Zone definer (6)

- *Security requirement for ZoneDEP, the data input program*
- special role
  - a user must have the ZoneDef role assigned
- must supply appropriate commandline parms or nothing works
  - this is a result of requiring the ZoneDef role as the first control
  - a connection must be made to the server to check the role which demands server info, userid etc.

Zone definer (7)

- *IMGLink block*

- was set up via INHERIT in B4; needs to be defined in the datamodel in B5

- holds the info about which image is associated with the record

Zone definer (8)

```
BLOCK TIMGLinkMana
 SETTINGS
  ATTRIBUTES = EMPTY,NODK,NORF
 BLOCK TSheetInfos
  BLOCK TSheetInfo
   FIELDS
    SheetName   : STRING
    SheetNr     : 1..20
    SheetStatus : 0..9
  ENDBLOCK
```

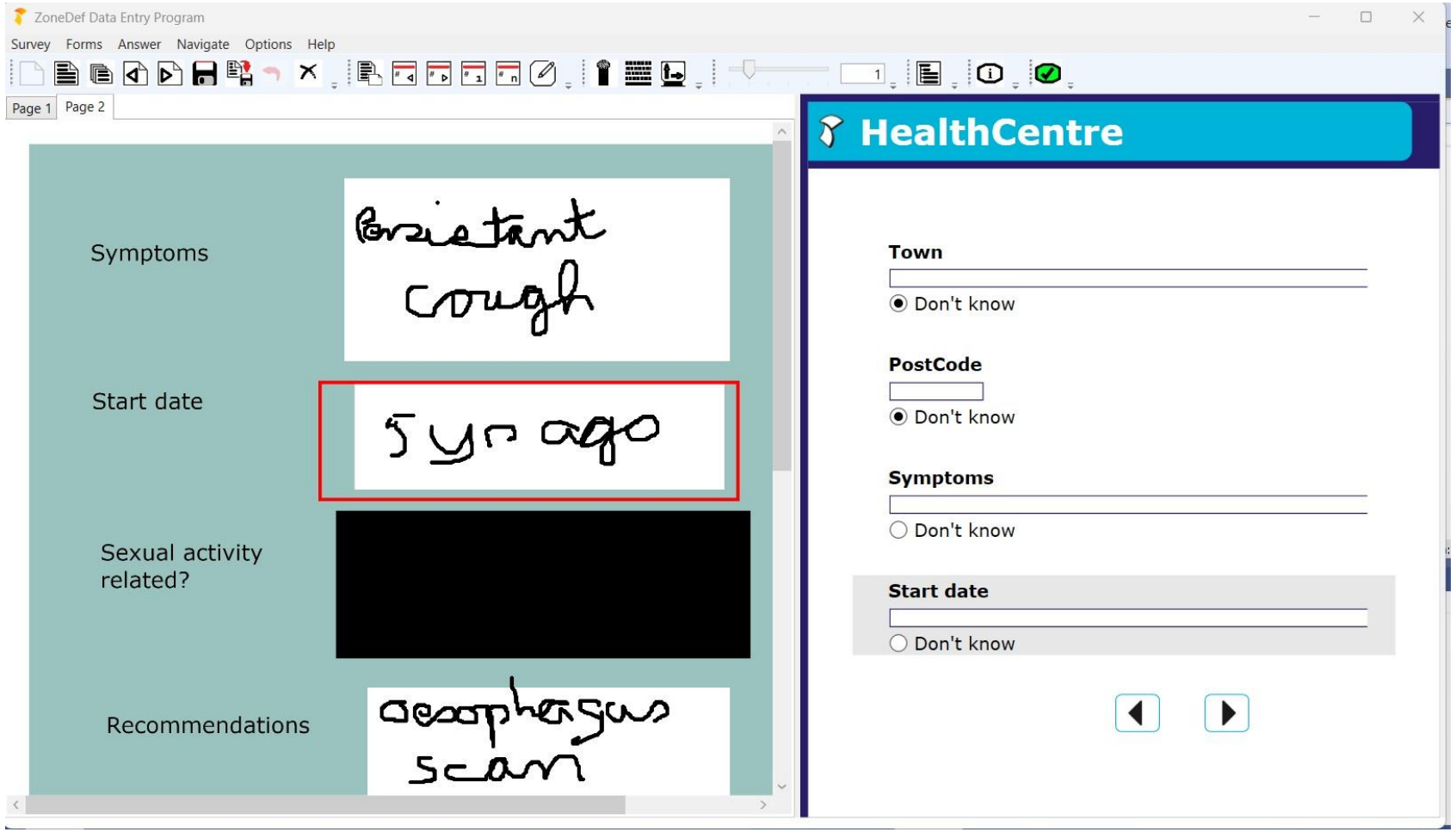
Zone definer (9)

```
FIELDS
    SheetInfo: ARRAY[1..20] OF TSheetInfo
 ENDBLOCK
 FIELDS
   SheetInfos : TSheetInfos
 RULES
   SheetInfos.KEEP
ENDBLOCK
```

Field definition

```
FIELDS
  IMGLinkMana: TIMGLinkMana
```

Zone definer (10)



Advanced programming trail

- As you have seen, these samples come in various degrees of difficulty
- You can follow the API samples with increasing difficulty and usefulness (C# mostly)
  - DataEntry, DataInterface, DataLink, DataRecord, Meta
  - BulkUserAdd
  - CustomClockWpf, SpeechWpf, PyCustomWpf (IronPython)
  - SurveyManagement (MVVM VB.Net)
  - AuditWorkshop
  - MenuWpf
  - Grid data editor
  - Data Entry Verification
  - Zone definer
  - Data bridge (if it’s finished)