



# Some different uses of expressions with Blaise 5 resource databases

Boris Allan, Arthur Menis, and Yelena Beale  
Westat, USA

IBUC – April 2025  
IBUC 2025, Rotterdam, The Netherlands



# Intro

Expressions fine tune what a properties' value will be.

- Screen layouts and their customization are part of the power of Blaise 5
- Customizing templates used in layouts involves setting properties.
- Often the properties are assigned a value – whether a literal value or a style – but Blaise 5 allows for Conditional logic and constructing an expression that will resolve to a value at runtime.

# Editor

## Expression language and the expression editor

- Expressions are context sensitive to the type of template and the property to be assigned a value
- Syntax is enforced
- Blaise 5 Expression help will point out the various variables available.
- These variables are organized to work with the sort of object on the page or template being designed.

# Variables

**Expression  
language and  
the expression  
editor**

- This paper focus on State and Page variables utilized in expressions
- For State, think what is actively going on these variables will help identify this
  - Example: `State.ActiveFieldName`
- For Page, think MasterPage and objects on the page
  - Example: Label and its Text property

# Language

## Expression language and the expression editor

- A set of functions for use in an expression
- If-Then-Else construct makes for flexible, powerful expressions
- The thing to know is expressions are meant to return some sort of value.
- For perspective, the next slide will be from Blaise 5 help and provides a list of the *types* of variables available.

# Blaise 5 Help: Expression Editor - Variables

## Variables

The expression editor gives you access to properties of all kinds of objects like *Server*, *State*, *DataModel*, *Client*, *Page*, *Group*, *Field*, *Control*, *SysDate*, *SysTime* and *Parallel*. Template [parameters](#) of type *General* can be used here as well and all Fields referred by [Field References](#).

- + [Server variables](#)
- + [State variables](#)
- + [Datamodel variables](#)
- + [Page variables](#)
- + [Client variables](#)
- + [User variables](#)
- + [Group variables](#)
- + [Field variables](#)
- + [FieldDefinition variables](#)
- + [Control variables](#)
- + [Skiplink variables](#)
- + [SysDate variables](#)
- + [SysTime variables](#)
- + [Parallel variables](#)
- + [Standard collection Methods and Properties](#)

## Side Note

**Expression  
language and  
the expression  
editor**

- In addition to Template properties, Templates may have Events configuration capability
- Event actions may require properties or conditional logic – and so expressions may be built here.

# Variables

## Expression language and the expression editor

- Alongside variables, standard methods and properties exist
- Methods allow for filtering the collection of items (e.g., the list of fields on the page or the list of categories for a given field)
- Method filtering also examine whether a given field `IsVisited` or its origin (is on the page or is a field reference)
  - e.g., `item.Origin = Reference` filters for `FieldReferences`

# Building an expression example: Text for a label (1)

- On a roster table, we have a label with a yellow background. We will construct the text contained therein via an expression.
- The motivation for the text is to provide context to someone such as a data manager to see what the active field is (what field has the focus in the screenshot to the right.)

The screenshot shows a web application window titled "CAPI roster with identifier information". The window has a blue header bar with the title and a "ENG" button. Below the header, there is a yellow highlighted area containing the following text:

```
NameGrpRoster  
LocalName: GrpRoster  
ROW ID: Brian  
COLUMN HEADER (Role=Description): Gender  
ActiveFieldName: Roster[2].Gender  
Page Name: Roster[2].Gender  
Page LocalName: Gender
```

Below the yellow area, there is a question: "What is Brian's gender?". Underneath the question is a table with the following columns: Identifier, DOB, Age, Gender, and Marital Status. The table contains three rows of data:

Identifier	DOB	Age	Gender	Marital Status
Alice	3/2/1998	35	Female	DIVORCED
Brian	12/12/1985	39	Male	SEPARATED
Charles	4/23/2020	4	Male	NEVER MARRIED

The "Male" value in the Gender column for Brian is circled in red. At the bottom of the window, there are two navigation arrows: a left arrow and a right arrow.

# Building an expression example: Text for a label (2)

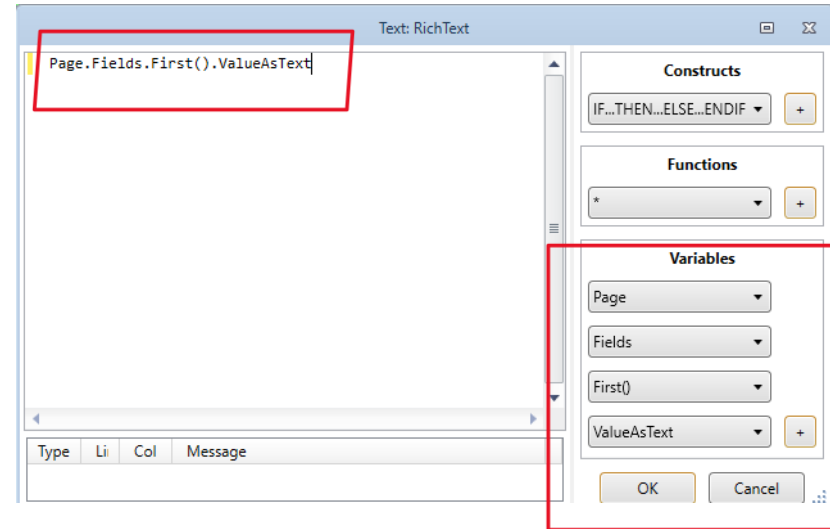
The screenshot shows the Visual Studio IDE. On the left is the Solution Explorer tree view, and on the right is the Properties window. The tree view shows a hierarchy: Master Page Template > Cell 0,0 > backgroundGrid > Cell 0,0 > mainGrid > Cell 0,0 > pageHeader > Cell 0,1 > buttonsGrid > Cell 0,0 > IdentifierInfo > Cell 0,0 > IdInfo. The 'IdInfo' property is highlighted in orange. A red arrow points from 'IdInfo' to the 'Label' property in the Properties window. The Properties window shows the 'Label' control with various properties. The 'Text' property is highlighted with a red box. The 'Text' property value is a green box with a grey setting icon.

- Set the Text property – click on the grey setting 

```
Page.Fields.First().ValueAsText
```

# Building an expression example: Text for a label (3)

- Select from the drop downs under Variables
- Click on the plus sign to add to the Text Window pane:
  - Page.Fields.First().ValueAsText
- The goal is to select the first field on the page that is for the Identifier column. A filter condition needs to be in the paren of First()



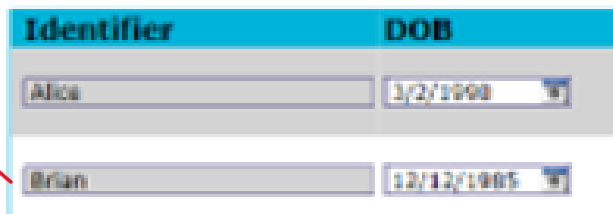
## Building an expression example: Text for a label (4)

- We want the Field object where the Field LocalName is Identifier.
- Of course, if we know the LocalName, we still need the full Block path.
- However, we do know all the fields on the roster are on the same block.
- State.ActiveFieldName is the fully qualified name of the field that has focus → This name has the full “block” path
- The hack is to replace the Active Field’s LocalName with ‘Identifile’ and then obtain the Identifier column’s value with the ValueAsText property

## Building an expression example: Text for a label (5)

- If ActiveFieldName is Roster[2].Gender, then replace 'Gender' with 'Identifier' and we get Roster[2].Identifier
- Given Identifier is the name of the person on the roster, the expression for the first line of display reduces to: Roster[2].Identifier.ValueAsText.
- In the example roster screenshot, the ValueAsText is Brian and that is what is displayed

ROW ID: Brian



Identifier	DOB
Alice	3/2/1990
Brian	12/12/1995

# Building an expression example: Text for a label (6)

## Snippets of the expression editor – identifying the desired variable and its property

- Expression editor for constructing other items for the label:
  - Column header value
    - Use.getRoleText('Description')
  - ActiveFieldName
  - LocalName of the active field

**Variables**

Page ▾

Fields ▾

First() ▾

GetRoleText() ▾ +

**Variables**

Page ▾

ActiveField ▾

LocalName ▾ +

**Variables**

Page ▾

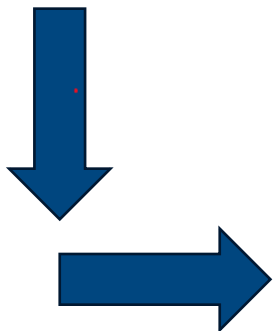
ActiveField ▾

Name ▾ +

# Building an expression example: Text for a label (7)

## The IDInfo Label

**ROW ID: Brian**  
**COLUMN HEADER (Role=Description): Gender**  
**ActiveFieldName: Roster[2].Gender**  
**Page Name: Roster[2].Gender**  
**Page LocalName: Gender**



## IDInfo Text property expression:

```
'ROW ID: ' + Page.Fields.First(item.name =  
                                REPLACE(State.ActiveFieldName,  
                                        Page.Fields.First(item.name = State.ActiveFieldName).localName,  
                                        'Identifier'  
                                )  
                                ).valueastext  
+ '<newline>COLUMN HEADER (Role=Description): '  
  + Page.Fields.First(item.name = State.ActiveFieldName).getroletext('Description')  
+ '<newline>ActiveFieldName: ' + State.Activefieldname  
+ '<newline>Page Name: ' + Page.Activefield.Name  
+ '<newline>Page LocalName: ' + Page.Activefield.LocalName
```

# An expression involving Groups: Looping on a roster – Group (OtherSpecify)

How do we display group role text (where Role=Example) in the source code below?

Note: Code below is edited for space reasons

```
GROUP grpRace
    {Group text} "Race information"
    EXAMPLE "Race information for person ^pOrder - ^{Identifier}"
    /"Race group"
FIELDS
    CodeAllRace (Enum01) "... "/ "Race" : tRace
    OthRaceOS (Enum01_OS)          ENG "SPECIFY: OTHER RACE"
    MAIN "CodeAllRace"
    OS "91"
    WATERMARK "OTHER RACE"      / "Race Other Specify"
    : tOtherSpecify, DK, RF, EMPTY//TString35_OthRaceOS//, RF, DK
ENDGROUP
```

# Looping on a roster – Group (OtherSpecify): Display information about the iteration of the loop.

CAPI roster with identifier information

**CAPI roster with identifier information** ENG

Race information for person 2 - Brian  
NameRosterRace[2].grpRace  
LocalName: grpRace

For this survey, Hispanic origins are not races.  
What is Brian's race? Please select one or more of the categories on card X-3.  
ENTER ALL THAT APPLY.

For this survey, Hispanic origins are not races.  
What is Brian's race? Please select one or more of the categories on card X-3.  
ENTER ALL THAT APPLY.

WHITE

BLACK OR AFRICAN AMERICAN

AMERICAN INDIAN OR ALASKA NATIVE

ASIAN INDIAN

CHINESE

FILIPINO

JAPANESE

KOREAN

VIETNAMESE

OTHER ASIAN

NATIVE HAWAIIAN

GUAMANIAN OR CHAMORRO

SAMOAN

OTHER PACIFIC ISLANDER

OTHER SPECIFY

Rather not answer

Don't know

# Looping on a roster – Group (OtherSpecify): Display information about the iteration of the loop.

- In the sample code, Race and its Other Specify is coded as a Group.
- In the expression editor, Group variables become available if one applies a “Group” type template such as OtherSpecify or Section.
- Given a roster array of groups, we make use of Group role text where the role name is “Example” Groups have a full Blockname path and a local group name.
- In this example, we identify the person’s name and the index of the array we are looping on:

```
Race information for person 2 - Brian  
NameRosterRace[2].grpRace  
LocalName: grpRace
```

# Looping on a roster – Group (OtherSpecify): Display information about the iteration of the loop.

- To display the three lines, we created a label on a pageheader template part and created an expression for the Text property of the label:

The image shows two parts of a software interface. On the left is a 'Template Part' window with a tree view. The tree view shows a hierarchy: 'Template Part' (expanded) contains 'headerGrid' (expanded), which contains 'Cell 0,0' (expanded) containing 'headerPanel', and 'Cell 0,1' (expanded) containing 'grid1' (expanded) containing 'Cell 0,0' (expanded) containing 'GroupLabel'. The 'GroupLabel' node is highlighted in orange. A red arrow points from this node to the 'Text' property in the 'Properties' window on the right. The 'Properties' window shows the 'Label' component with various properties. The 'Name' property is set to 'GroupLabel' and is highlighted with a red box. The 'Text' property is also highlighted with a red box and contains a green field with an expression editor icon.

Template Part

Name: gjba\_PageHeaderWithLangSel

- Template Part
  - headerGrid
    - Cell 0,0
      - headerPanel
    - Cell 0,1
      - grid1
        - Cell 0,0
          - GroupLabel

Properties Events

Label

Margin 2

Name GroupLabel

RowIndex 0

RowSpan 1


ScreenReaderText

ScreenReaderTextK

ScreenReaderTextSi None

Text

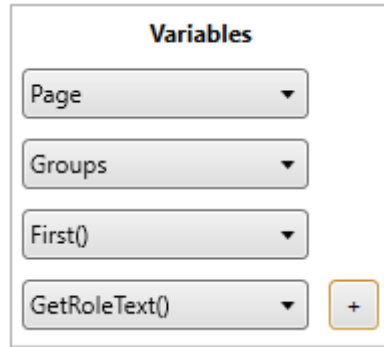
## Looping on a roster – Group (OtherSpecify): Display information about the iteration of the loop.



**Race information for person 2 - Brian**  
**NameRosterRace[2].grpRace**  
**LocalName: grpRace**

```
Page.Groups.First().getRoleText('Example')  
+ '<newline>Name' + Page.Groups.First().name  
+ '<newline>LocalName: ` +  
Page.Groups.First().localname
```

## Looping on a roster – Group (OtherSpecify): Display information about the iteration of the loop.



Page.Groups.First().getRoleText('Example')

Here .First() has no filter, so the first group on the page is identified.

The Example Role text is display.

***Page.Groups.First().getRoleText('Example')***

+ '<newline>Name' + Page.Groups.First().name

+ '<newline>LocalName: ` +

Page.Groups.First().localname

# Looping on a roster – Group (OtherSpecify): Display information about the iteration of the loop.

```
Page.Groups.First().getRoleText('Example')
```

```
+ '<newline>Name' + Page.Groups.First().name
```

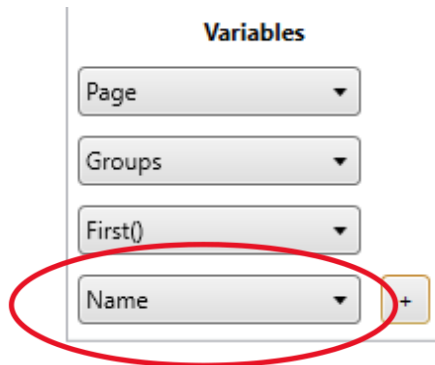
```
+ '<newline>LocalName: ` + Page.Groups.First().localname
```

- Like in Fields, `.name` produces full block and localname and `.Localname` returns just the short name – the group name, in this case

Variables

Page	▼
Groups	▼
First()	▼
Name	▼

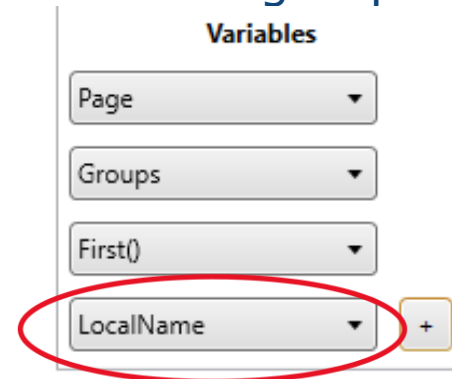
+



Variables

Page	▼
Groups	▼
First()	▼
LocalName	▼

+



# Conclusion

- ❖ Expressions are powerful, but they can be a challenge.
- ❖ The on-line help is a good start, but more examples will always be helpful.
- ❖ Having a sense of the metadata may be helpful, especially in the sense of the type of object (e.g., category vs. field vs. group, etc.) that one is trying to process.
- ❖ Being aware of how the instrument is programmed, with elements like role text and groups, is also important

# Thank you

Let us know if you have any questions.

[westat.com](https://www.westat.com)



Photos are for illustrative purposes only. All persons depicted, unless otherwise stated, are models.