Expressions in Blaise 5 Resource Editor

Nikki Brown and Yelena Beale, Westat

1. Abstract

This paper will examine ways in which the Expression Editor in Blaise 5 can be used to link specific layout objects to templates in order to create a layout instruction. Expressions created in Blaise can be very straightforward or incredibly complex, each providing powerful capabilities to the survey designer.

Expressions are created by utilizing the drop-down lists with functions and variables each editor provides in its current context. We describe our experience learning to use the Expression Editor to meet client requirements.

2. Overview of Expression Editor

"An expression is a content aware combination of explicit values, constants, variables, operators, and functions that are interpreted and evaluated at runtime to produce a value. With expressions you can change colors of active controls, set visibility of controls and many things more." - Blaise Help File

The Expression Editor in Blaise 5 is a powerful tool that enables a developer to accomplish many tasks that cannot be accomplished within the Blaise code. Expressions have been used for everything from toggling buttons off and on to security. There is information on the Expression Editor in the Blaise Help file and there are also many Expressions already written in the default templates that come with the default Resource Editor and in the Resource Editors of the samples. Reviewing the Expressions that have already been created can give the developer ideas as to what they themselves can do. Why does error text only show when there is an error? Take a look at the default Vertical template's Expression for the error grid's Visibility Property to find out. (Fig. 1)

Fig. 1. Visibility Expression from Error Grid

Visibility: Visibility

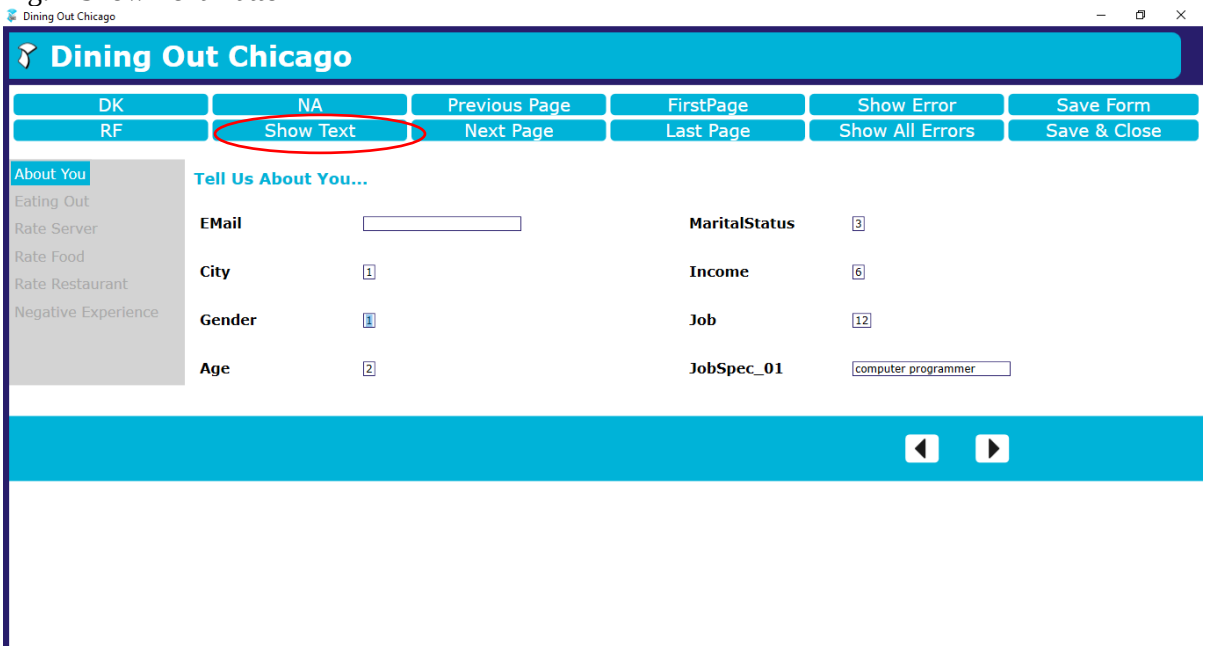
```
IF Field.Errors.IsRelevant THEN
  IF (Field.Errors.FirstRelevant.ErrorKind <> Route OR
(NOT State.RuntimeSettings.SolveErrorsImmediately)) OR
(State.PreviousPageIndex = Page.Index AND
State.NavigationDirection = Forward) THEN
    'Visible'
  ELSE
    'Collapsed'
  ENDIF
ELSE
  'Collapsed'
ENDIF
```

3. Common Usages of Expressions

For this paper we are going to examine uses of Expressions in button controls. In the OnClick property of a button control, there are a number of actions available. Some such as FirstPage, LastPage and Save need no further help to accomplish their task. Others, such as Assignfield, will require at least another step. If the Action you have selected requires more steps, the Editor will show the Properties on the right side of the screen to let you know other work is needed. For the Assignfield action, you need to tell what field is to be assigned and what value you want to assign. Remember that if the field you are assigning is in the Blaise code, you need to add it as a field reference in the Resource Editor.

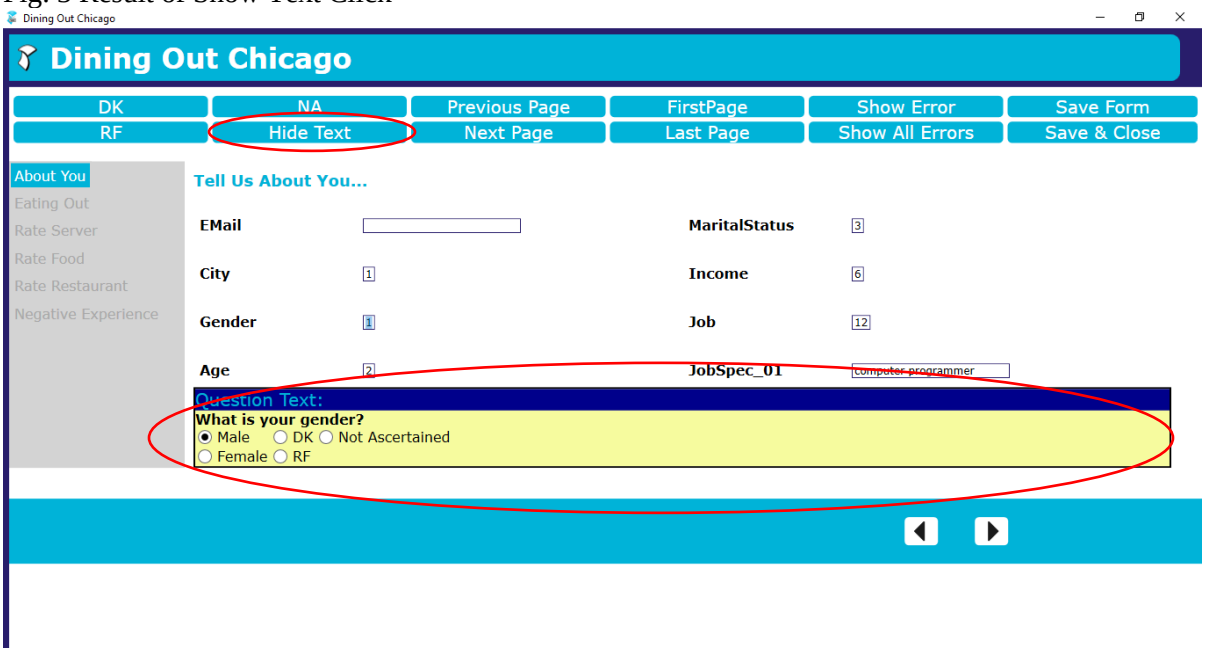
In this example, we are talking about the 'Show Text' button. (Fig. 2) To have all the pieces work in tandem, we used Expressions for both properties, (Text and Visibility) and events (OnClick). In this picture, the button has not been clicked, so no Question text is showing on the screen.

Fig. 2 Show Text Button



When the button is clicked, a couple of events take place. The text on the button changes to 'Hide Text' (Fig. 3) and the Question text and Answer categories show up at the bottom of the screen for the Active Field Gender.

Fig. 3 Result of Show Text Click



In the OnClick, event we have added a Conditional that uses Procedure Calls to change the value of the newly created Server Variable from its default of 'False' (Fig. 4) to 'True' (Fig. 5) and back again. The Server Variable needs to be set back to 'False' in the Else statement otherwise the button will not return to its initial state. (Fig. 6)

This Server Variable can then be used in other Expressions to toggle off and on, Visibility and Text.

Fig. 4 Server Variable ShowText default False

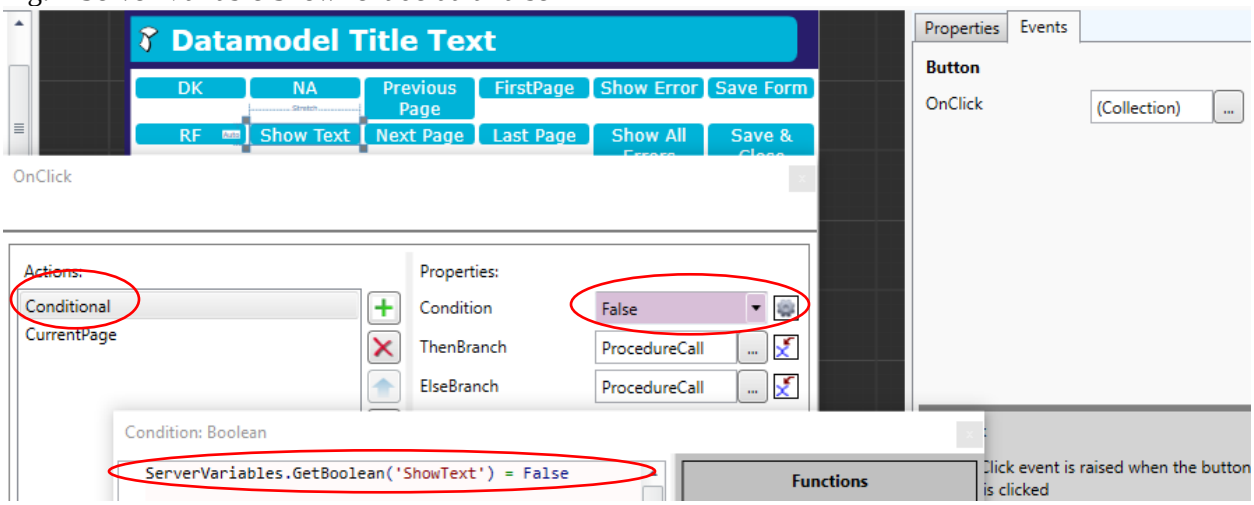


Fig. 5 Setting Server Variable to ‘True’ in the Then Branch

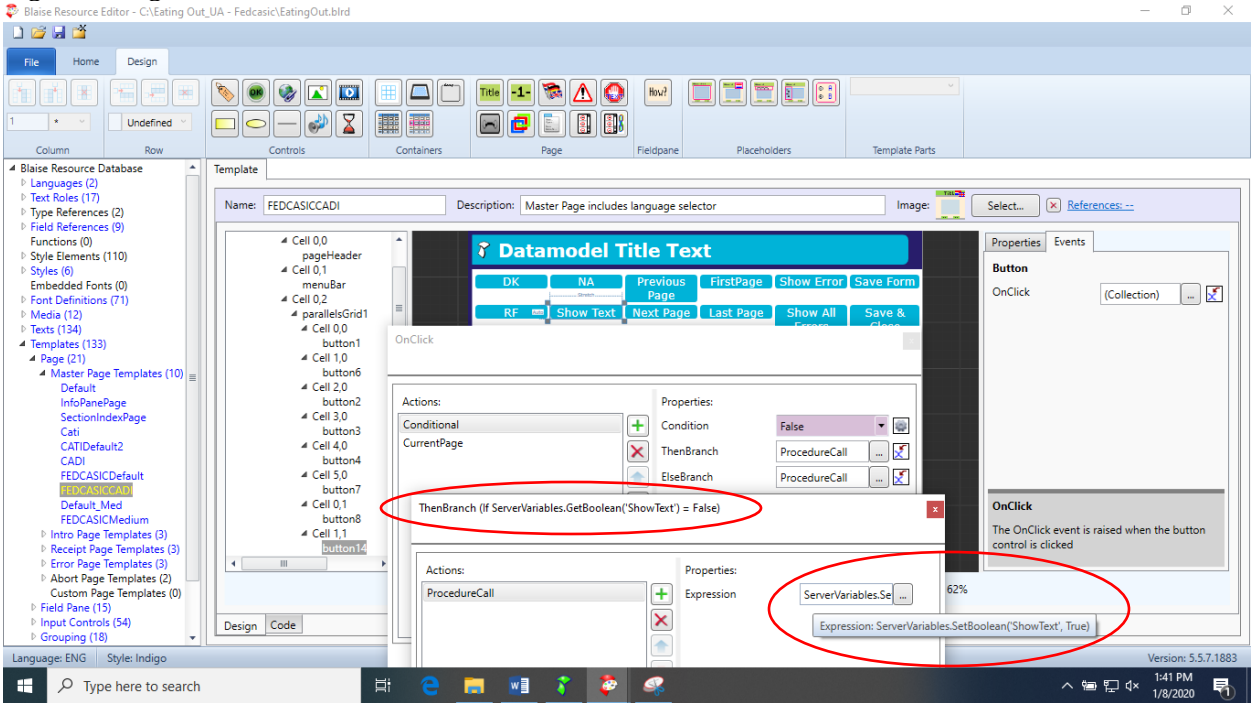
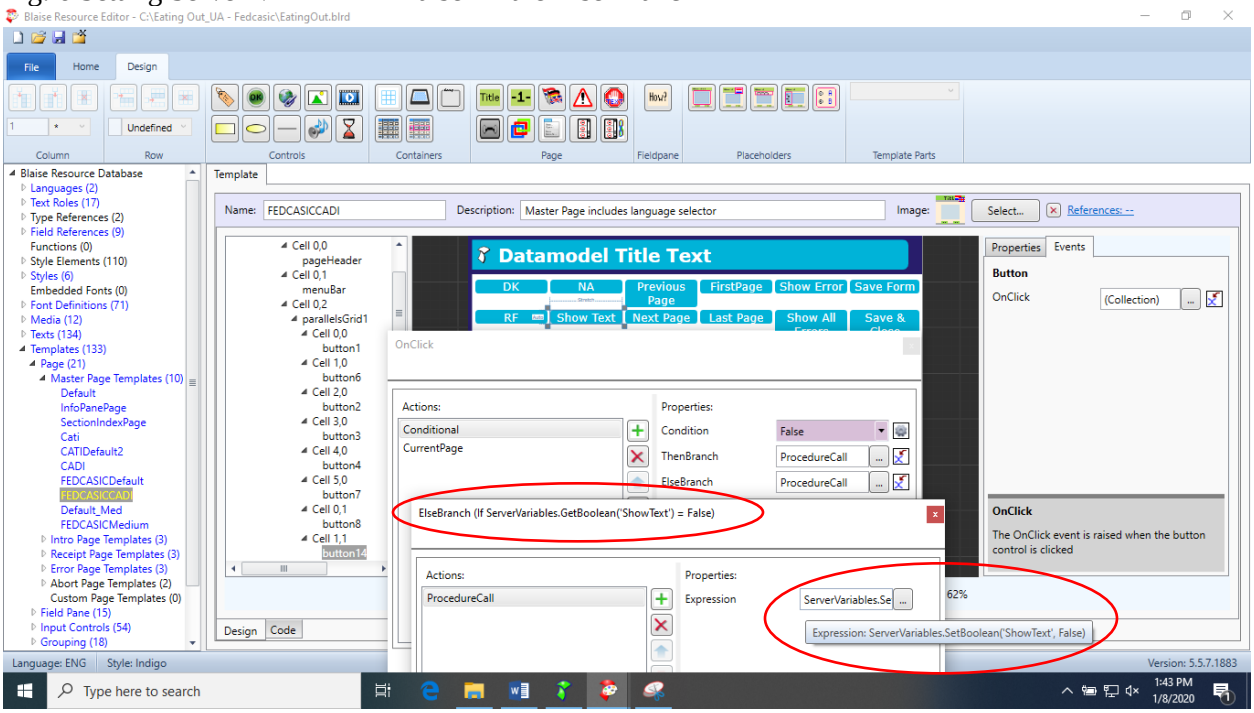
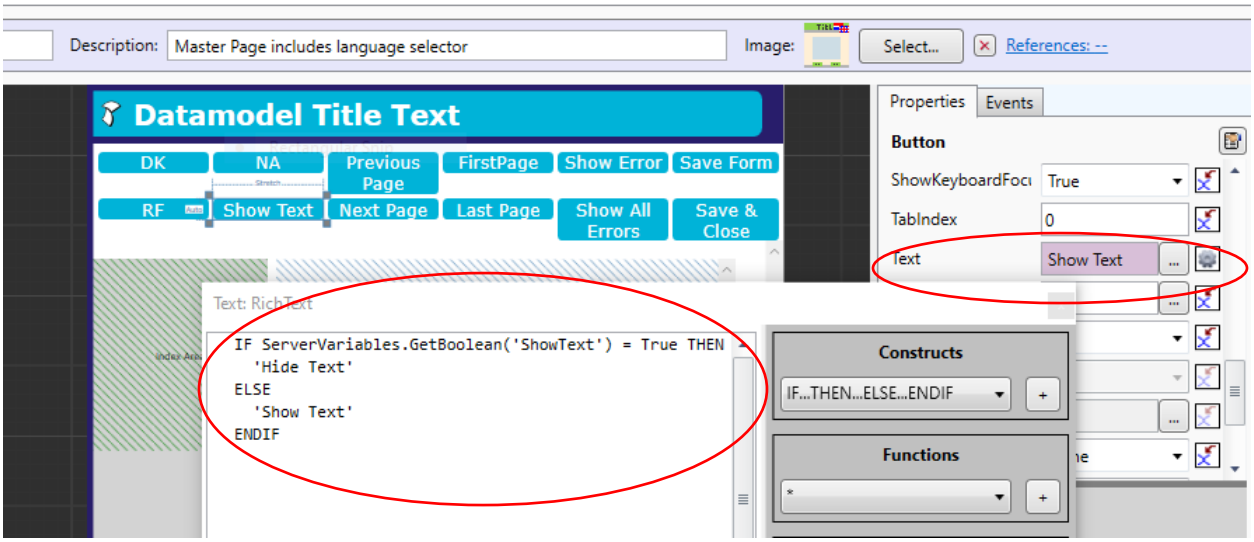


Fig. 6 Setting Server Variable to ‘False’ in the Else Branch



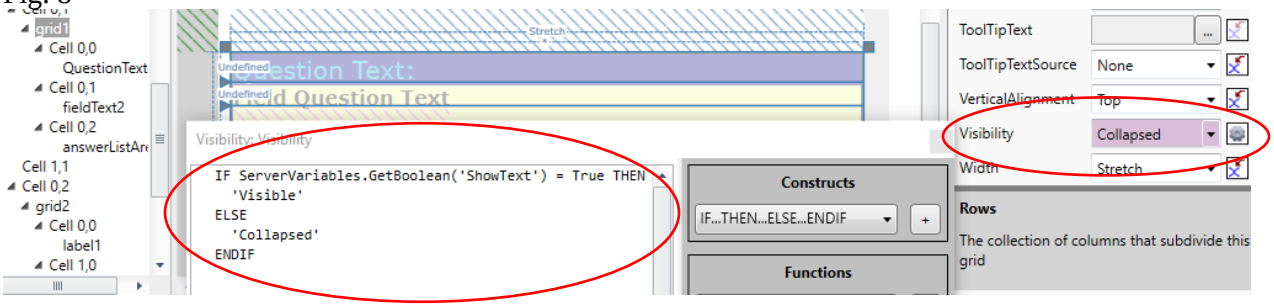
Here we refer to the Server Variable to pick which text to display on the button. (Fig. 7)

Fig. 7



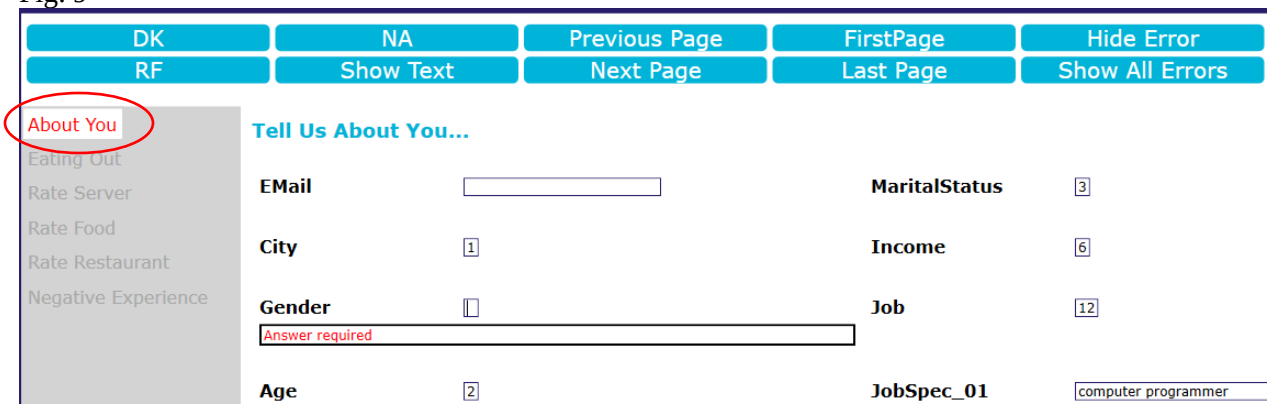
Here we refer to the Server Variable to toggle off and on the Visibility of the grid that displays the Question Text and Response Values for the active field. (Fig. 8)

Fig. 8



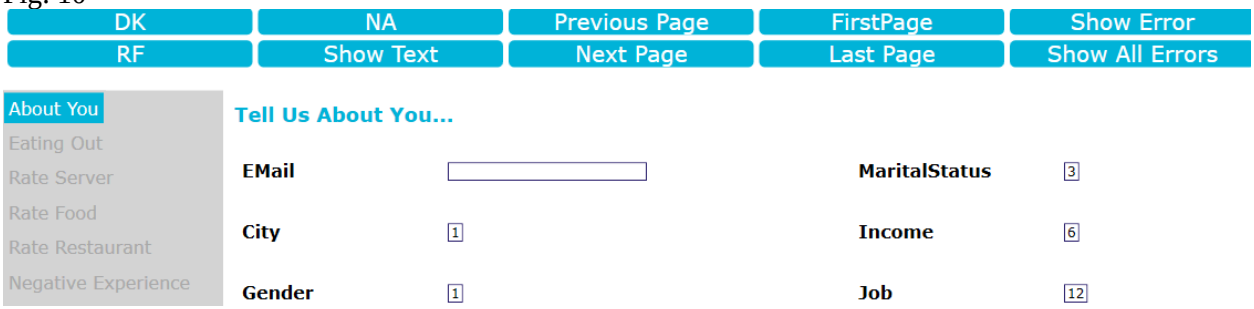
The ‘Show’ Error button used below went a step further and when clicked, changed the Font Color and Background of the Section Header in the IndexItem template. (Fig. 9)

Fig. 9



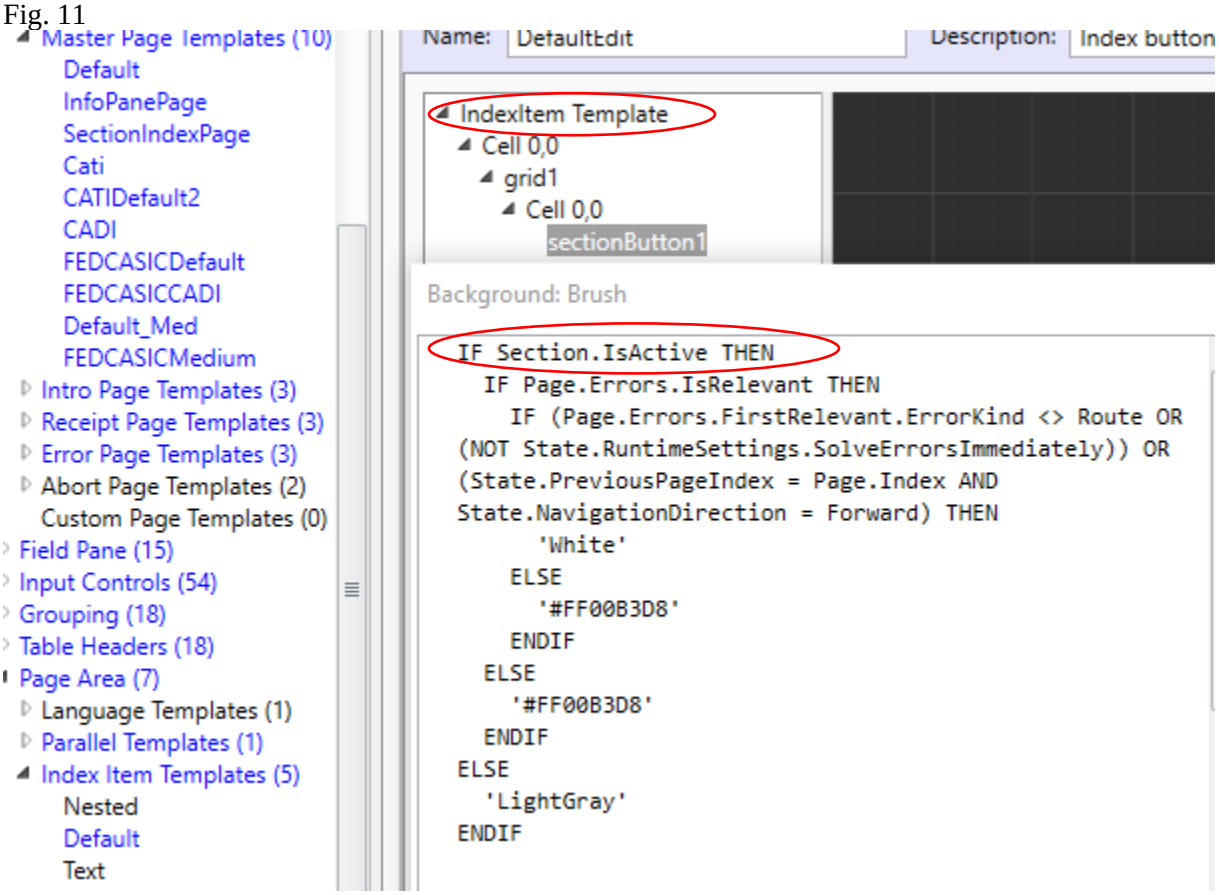
When the error was fixed by entering a value for the ActiveField Gender, all the changes were returned to their original state. (Fig. 10)

Fig. 10



In the IndexItem Template we used the same Expression from the default resource editor that controls the Visibility of an Error control for changing the Background and Font Colors of our Section Headers. We added “If Section.IsActive” to the Expression so that an error in one section wouldn’t change the colors of all sections. (Fig. 11)

Fig. 11



If this Expression had been copied to a different control, an error (Fig. 14) could occur because the controls do not have the same lists of functions and variables available to them. The Expression Editor is context aware. A button placed in the IndexItem template has a choice of Section available (Fig. 12) but in the Vertical template it does not. (Fig. 13)

Fig. 12 Index Template Section top of list

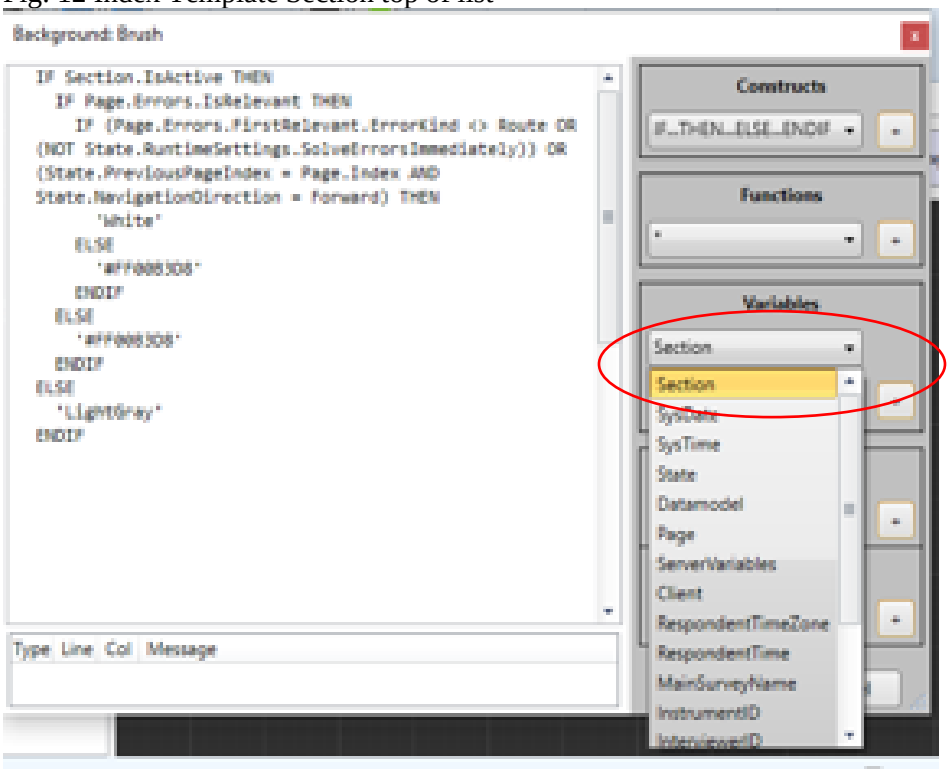


Fig. 13 Vertical Template State top of list (ie no Section)

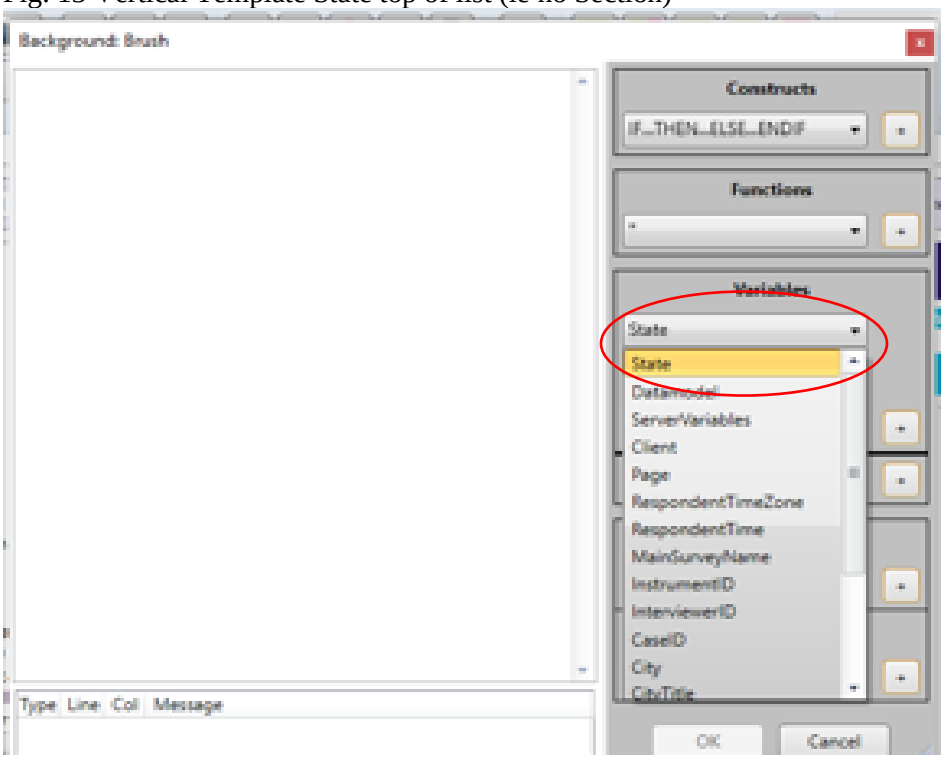
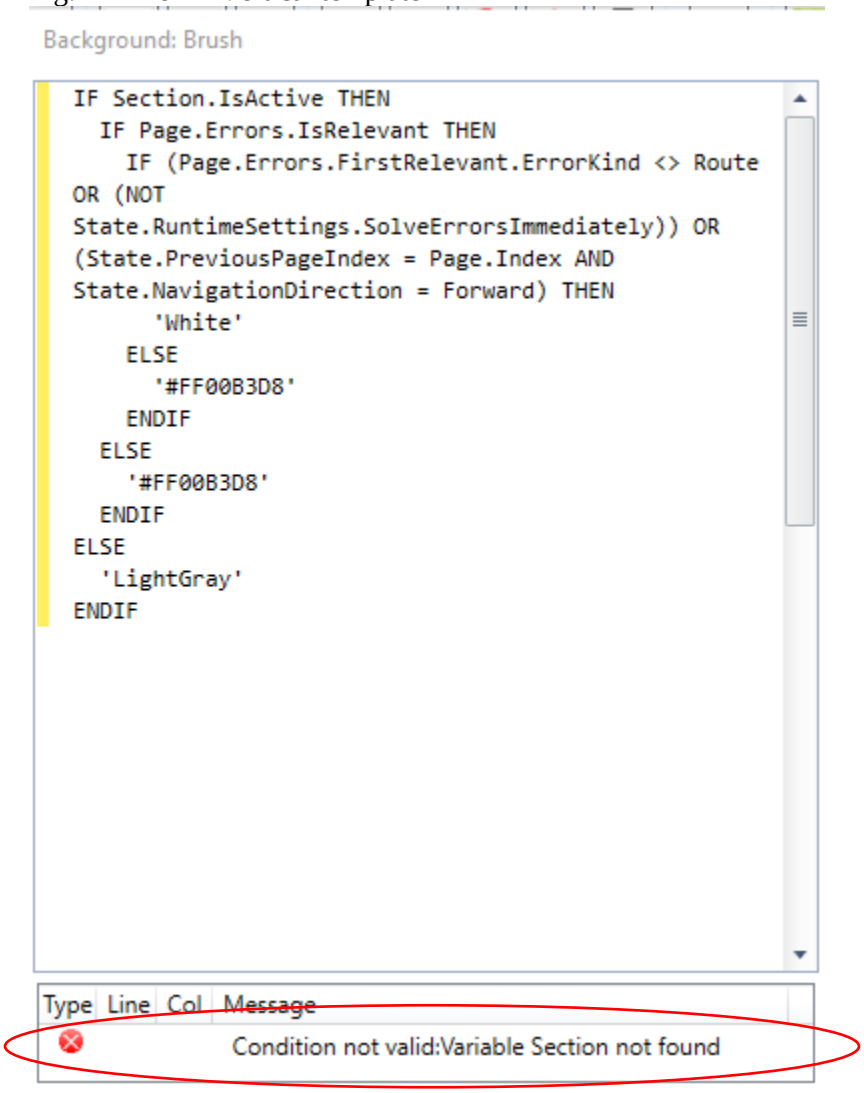
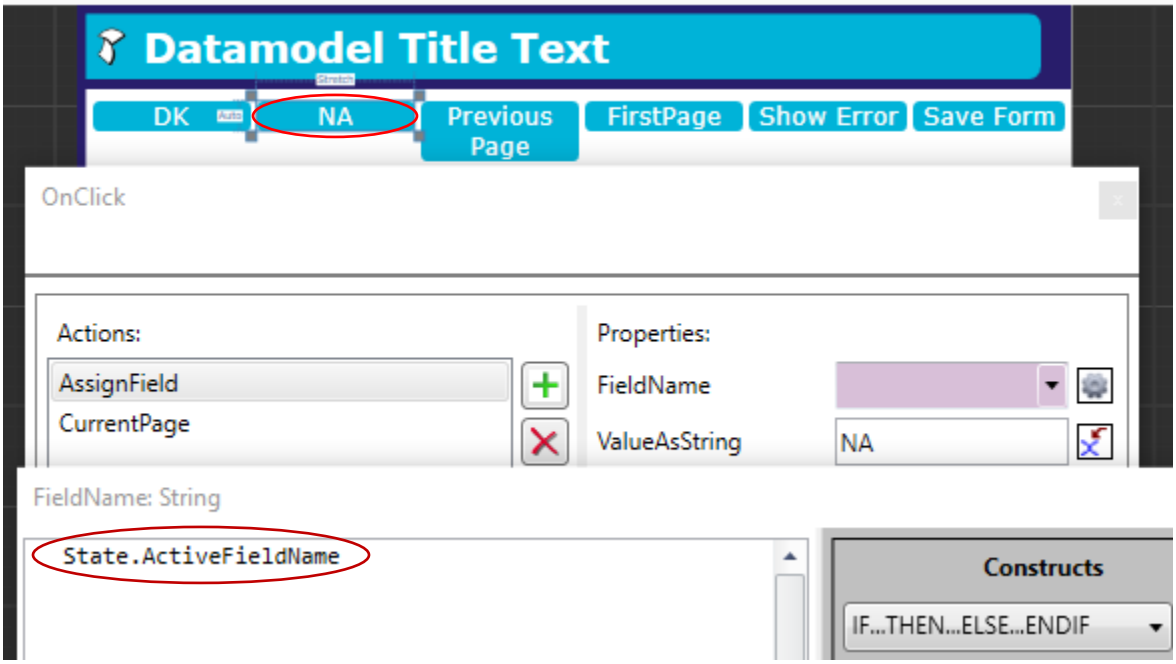


Fig. 14 Error in Vertical template



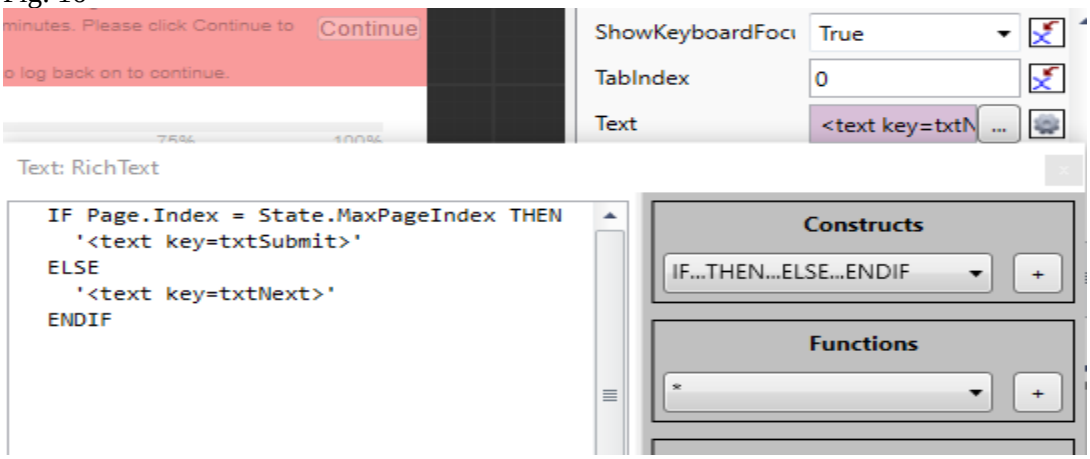
In this multimodal demo survey, the self respondent mode was allowed to leave fields empty. In editing mode most fields were required. We created a special attribute "NA" for Not Ascertained that editors can use to fill unanswered questions. The NA button uses an AssignField action. (Fig. 15) Since the field is actually on the route, it does not have to be added to the field references which would be cumbersome if all database fields needed to be added and mapped. The Expression just uses the State.ActiveFieldName for the assignment.

Fig. 15 NA attribute assignment



Another common use of an Expression is for setting the text of the Next button to Submit if on the last page of the survey. (Fig. 16)

Fig. 16



Off route fields such as statuses or time and date stamps, need to be added to the field references (Fig. 17) and mapped (Fig. 18) if values are going to be assigned to them in an expression. Remember that you have to map them if your Field Reference doesn't match the database exactly. If the database field is not at the .blax level (ie in a sub block) it won't match because it needs the block name attached to it.

Fig. 17 Field References

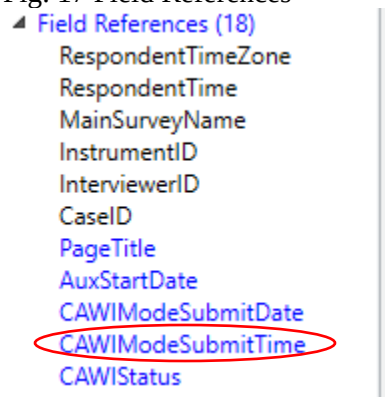
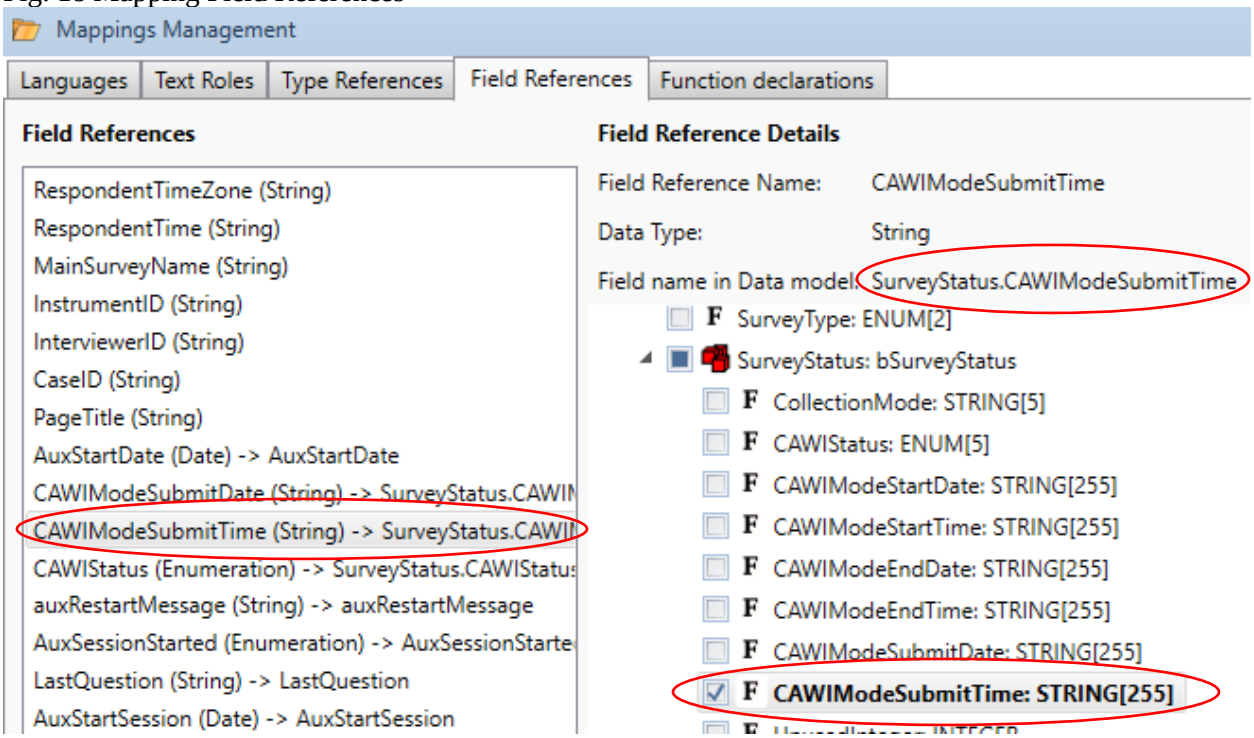
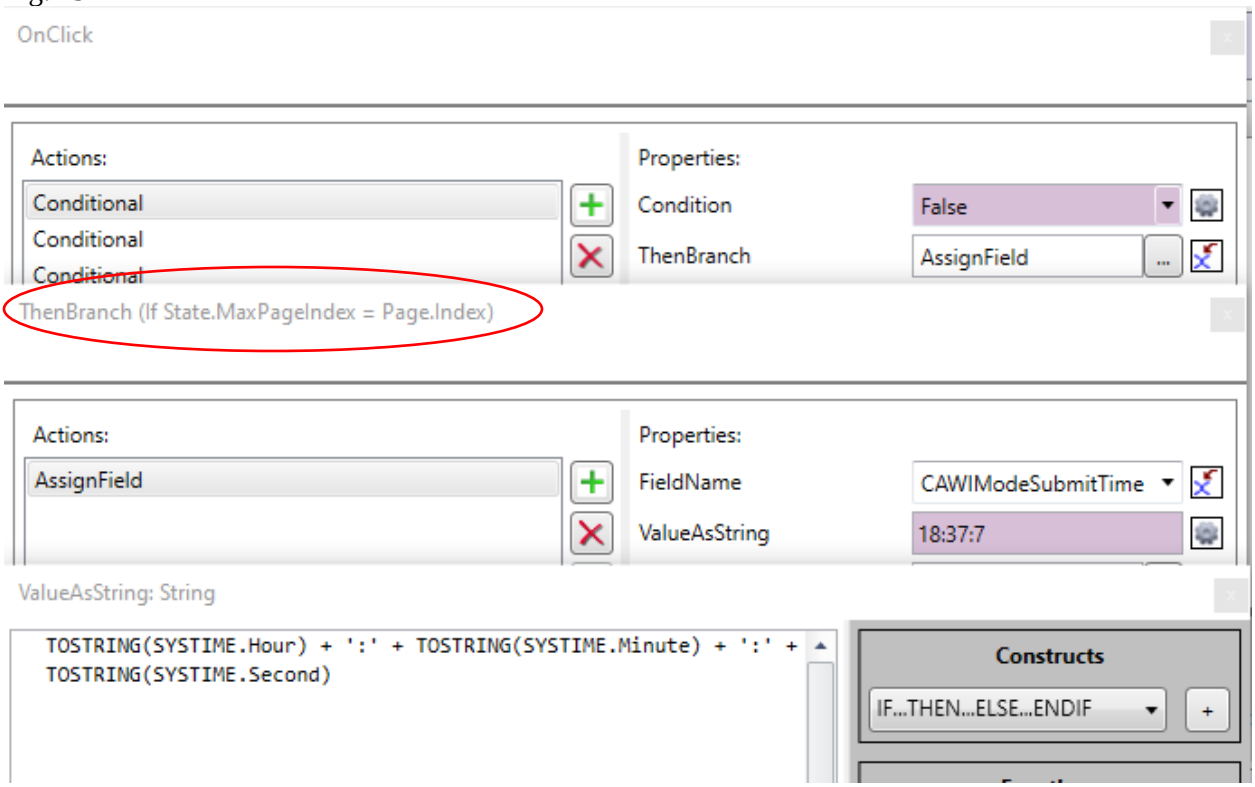


Fig. 18 Mapping Field References



After mapping the fields, you can make the assignments. In this case we assigned based on the same condition used for the Submit text. (Fig. 19)

Fig. 19



4. Conclusion

This paper examined some of the many ways that the Expression Editor can be used to support your survey using button control examples. It is a powerful context aware tool with broad capabilities that is evaluated at runtime to support survey functionality.

Even if you haven't written any yourself, your surveys are using the Expressions that come in the default templates. As we all learn in different ways, remember that there are many examples in the Resource Editors that come in the Blaise Samples and the default templates. If you do something novel with an expression, please share it with your organization so everyone can benefit from it. If you encounter this message frequently when learning to work with expressions -- "An unrecognized error has occurred" -- try building the expression one step at a time to locate the error. P.S. Don't forget to have fun with them.