Adventures in Groovy – Part 14: Returning Errors (Form Cells)

Introduction

To expand on Part 13 of this series, which covers stopping a form from saving when there are validation errors, is identifying the errors by cell and communicating with the user the problems at a cell level. This does NOT stop at the first error and throw an exception. This will iterate through all the errors and explain each one at a cell level for the user to correct. The following example will use similar code and concepts, but will apply validations to each cell by changing the color and setting a tool-tip with the explanation of what the validation error is.

Before we continue, the methods to do this do not make use of the MessageBundle. I think this is a miss because one bundle can be reused for similar validation, and the current methods assume a single language. There is a way to use it indirectly. There is a bug that is causing issues with the method, so we will assume basic functionality and come back to the use of a MessageBundle when the bug is fixed

Throw an Exception (Interrupt Form Save)

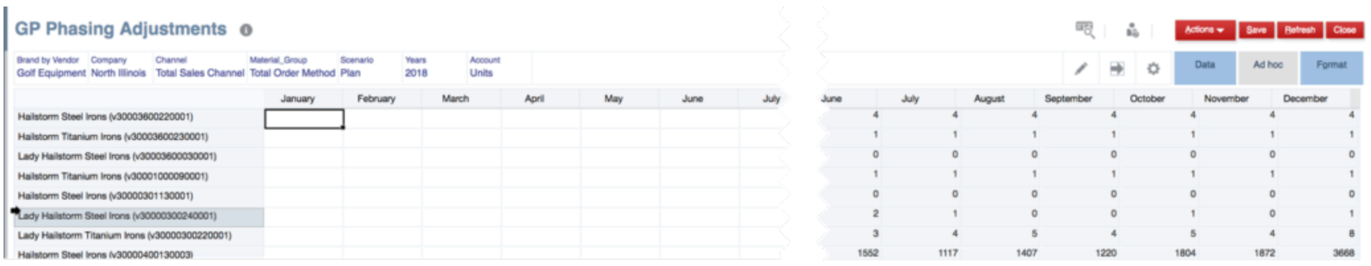
The basic inclusion of cell validation is very simple. As the code iterates and validates the cells, the following will change the background color, add a tool-tip, and invalidate the form and stop it from saving any data to Planning.

```
def BackErrColor = 16755370 //Red
it.addValidationError(BackErrColor, "error message here",false)
```

The color can be different for different errors and it completely customizable. The error message can be anything necessary.

Consolidated Example

The form associated to this rule has the ability to adjust a number by either increasing or decreasing the units by month.


GP Phasing Adjustments

Brand by Vendor	Company	Channel	Material_Group	Scenario	Years	Account
Golf Equipment	North Illinois	Total Sales Channel	Total Order Method	Plan	2018	Units

	January	February	March	April	May	June	July
Hailstorm Steel Irons (v30003600220001)							
Hailstorm Titanium Irons (v30003600230001)							
Lady Hailstorm Steel Irons (v30003600030001)							
Hailstorm Titanium Irons (v30001000090001)							
Hailstorm Steel Irons (v30000301130001)							
Lady Hailstorm Steel Irons (v30000300240001)							
Lady Hailstorm Titanium Irons (v30000300220001)							
Hailstorm Steel Irons (v30000400130003)							

June	July	August	September	October	November	December
4	4	4	4	4	4	4
1	1	1	1	1	1	1
0	0	0	0	0	0	0
1	1	1	1	1	1	1
0	0	0	0	0	0	0
2	1	0	0	1	0	1
3	4	5	4	5	4	8
1552	1117	1407	1220	1804	1872	3668

To illustrate this, here is an example of looping through cells and validating two things.

1. Units can’t ever be adjusted to a negative amount – they can be decreased, but never to a negative value.
2. Any change to units must be offset to have a full year impact of zero.

```
def BackErrColor = 16755370 //Red

def CaseTotal = it.crossDimCell('Jan').data +
it.crossDimCell('Feb').data + it.crossDimCell('Mar').data +
it.crossDimCell('Apr').data + it.crossDimCell('May').data +
it.crossDimCell('Jun').data + it.crossDimCell('Jul').data +
it.crossDimCell('Aug').data + it.crossDimCell('Sep').data +
it.crossDimCell('Oct').data + it.crossDimCell('Nov').data +
it.crossDimCell('Dec').data
operation.grid.dataCellIterator('Working_Inp','Jan','Feb','Mar
','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec').each
{
 if(it.data + it.crossDimCell('OEP_Working').data < 0.0)
   {
   def change = it.data + it.crossDimCell('OEP_Working').data
   it.addValidationError(BackErrColor, "Your adjustment forces
the new cases to be a negative volume. Increase your
adjustment by $change", false)
   }
 else
  {
  if(CaseTotal != 0.0 && it.data != 0.0)
     it.addValidationError(BackErrColor, "Adjustments must not have a full year impact. Currently, the data would change by $CaseTotal.", false)
  }
 }
```

Enhancement Request

One thing you might notice is the lack of inclusion of the messageBundle object. I have requested an enhancement, as it only makes sense that it be used here, and they have added it to the enhancement list. So, look for this be added in the future. It can be identified internally by the following.

Enh 27656951 – EPBCS – GROOVY FUNCTION ERRORING

I don’t know why, but Oracle has no way of getting the message based on the local from the messageBundle. Many of the methods, like getMessage, are not made available to us as developers, that would likely circumvent this issue.

Summary

As with the other validation methods, this introduces a huge benefit in both usability and budget accuracy. Any time data validation can be performed proactively, everybody wins. There is less of a burden on administrators and users get instant feedback they can easily and quickly fix.