Adventures in Groovy – Part 44: Don’t Waste User Time and System Resources

Data forms that have rules on them run when a user saves a form, regardless of whether the form has edited data. Sometimes this is by design, to let a user run a calculation. But, most of the time, it is an accident. There is no reason to use system resources and have the user wait for a calculation to finish when it doesn’t need to be executed.

Check If Data Has Changed

There is a very simple way to check if a data form submitted has data that has been edited, or changed. Several previous posts have talked about the data iterator. The data iterator is used to loop through the cells in a form. Predicates can be used to only loop through the cells that have been edited. The documented example shows the code to identify and loop through the edited cells, printing each to the job console.

```
operation.grid.dataCellIterator({DataCell   cell   ->
cell.edited}).each {
     println("$it.memberNames, cell data: $it.data")
 }
```

If you are starting to understand the methods and collections, you notice that this returns a list of data cell types. From this, I can get all the properties of each cell. Since this returns a list, I can also use the size method. The size method returns the number of elements.

```
operation.grid.dataCellIterator({DataCell   cell   ->
cell.edited}).size()
```

This will return the number of edited cells. It isn’t a stretch to take this one step further and wrap it in an if

statement to see if the number of edited cells is more than 0. If it is 0, then no cells have been edited.

```
IF(operation.grid.dataCellIterator({DataCell   cell   -> cell.edited}).size() == 0){
   ...take action
}
```

Inform The User Or Not

This is where there is some administrator preference. I can argue both ways, but I think it depends on the users and the expectations of what happens when a form is saved. One one hand, I can argue that the rule should exit without notifying the user that nothing ran. It is an extra click that they have to respond to that might be annoying. Telling them that nothing changed, and no calculation ran may not be needed. In this case, the calculation exits gracefully, and everybody moves on. Using return exits the calculation.

```
IF(operation.grid.dataCellIterator({DataCell   cell   -> cell.edited}).size() == 0){
    return
}
```

On the other hand, the user be expecting something to happen. They might need to know that they didn’t edit anything. If this is the situation, I can throw an exception and prompt the user there was an issue. Using throwVetoOperation will do just that. It initiates an error and nothing further runs.

```
IF(operation.grid.dataCellIterator({DataCell   cell   -> cell.edited}).size() == 0){
     throwVetoException{"No data was edited and no business logic was executed."}
}
```

The throwVetoException method can also use the Groovy message bundle and bundle loader if you are using those classes.

That’s A Wrap

I have used both methods, exiting without notifying the user and existing and notifying the user. My preference is to simply exit and not run or notify the user because normally they don’t need to know nothing ran because nothing changed. Regardless, I think it is a good practice to add this to your Groovy calculations to account for a user saving a data form when no data was edited.