Adventures in Groovy – Part 46: Start Making Rules More Reusable, Part 1

One thing that I have spent a lot of time on is making calculations independent of forms so that they can be used on any form. For example, rather than hard coding a script to look at a form with one column header and one row header, I am now building things to be dynamic so that the POV, the rows, and columns all are read dynamically and identifying the edited cells is independent of the source it is looking at. This will be a multi-post article because there is a lot to cover.

Think Different

I have not talked about some of the, what might seem like, less functional classes and methods in the API. Most of my examples don't go into their use. The reason is solely trying to break concepts apart and try to not overload people with everything at once. What if I told you you could eliminate most of your substitution variables? What if I said you only need one data map? What if I told you that you could use rule properties like never before? Spoiler alert, you can!

Building The Foundation

The first concept I want to share is a simple one, but it is the start of making your scripts dynamic. When working with grid builders, a reference has to be made to the plan type. This is a very simple thing but has to be changed for different plan types. You might first think to put this in a script and embed the script, but there is an easier way that makes it completely independent of the plan type or application.

Everything I have shown you with grid builders starts with this. This means that every rule has to have a hard-coded plan type in it.

```
Cube cube = operation.application.getCube("plantypename")
```

Well, this isn’t the case. It can be done without hard coding the play type name. The cube variable can be set by getting the cube that the rule is created in so the rule will work on any plan type in any application.

```
Cube cube = rule.getCube()
```

I can take that one step further and eliminate the cube variable all together.

```
DataGridBuilder builderSubmit = rule.getCube().dataGridBuilder("MM/DD/YYYY")
//or
DataGridBuilder builderSubmit = rule.cube.dataGridBuilder("MM/DD/YYYY")
```

Grid builders aren’t the only class that uses this. If you are doing anything with metadata, this will also benefit those scripts.

```
Dimension productDim = operation.application.getDimension("Period", cube)
// can be changed to
Dimension productDim = operation.application.getDimension("Period", rule.cube)
```

That’s A Wrap

We have access to all kinds of things that we can make use of now through these classes. The application class exposes the currency mode and the default currency. We have access to the smart lists and can access those. Could we use those in calculations? Maybe it is used as a map, where label and description is an account conversion between plan types? We can get the dimensions, so a calculation could see if a

dimension exists. Maybe we can dynamically create fix statements based on the dimensions in the cube (aggregate everything that is sparse). User variables can be set, so maybe if a calculation runs and the user variables aren't set, we ask the user to set it with an RTP, then continue the calculation?

The rule class has methods to get the description, the name, and the rule properties. I can't say I have done it, but maybe we used the description as a variable? Maybe we have the name in a convention that specific characters mean something and are brought into the rule, like a the scenario name?

Some of these are just thoughts, and some of them are things that we could implement and use. My point is that there are all kinds of things we have access to dynamically that we didn't before. The apps I am building don't have variables for the open periods anymore because I can get them dynamically in the calculation based on the scenario being calculated. No more start and end month. No more current month. And, if they are needed for reports or forms, have the calculation set them if they are wrong.

So, what are you thinking? Do you have something you have done that you couldn't do before Groovy? Share it by commenting.