Two GlideRecord Queries

Lab 7.1 20-25 minutes

Lab objectives

You will perform the following in this lab:

- Write a GlideRecord query to locate a set of Incidents.
- Write a second GlideRecord that:
  - Runs a query to locate a set of users
  - Triggers the Business Rule from one table but makes updates to records on a different table

A. Script 1: Find Active SAP Incidents

1. Create a new Business Rule.

   Name: **Lab 7.1 SAP Incidents**
   Table: **Incident [incident]**
   Active: **Selected (checked)**
   Advanced: **Selected (checked)**
   When: **after**
   Order: **150**
   Insert: **Selected (checked)**
   Update: **Selected (checked)**

2. Switch to the **Advanced** tab.

3. Within the ***executeRule()*** function in the Script field, type **vargr** followed by the <tab> key to insert the standard GlideRecord syntax *(stored in the baseline Syntax Editor Macro)*.

4. Select the **Format Code** icon on the Syntax Editor toolbar to properly format the script.

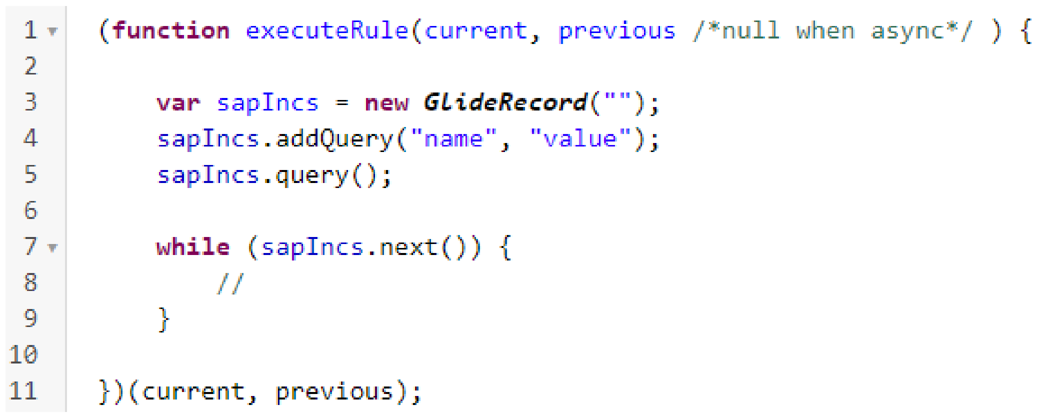
5. Update the Script Macro to suit your current requirements.

   a) Update the name of the GlideRecord object to something more meaningful (considered good practice).

   b) Replace the word "*if*" with **while**.

   c) *Optional: Insert a blank line between the query() statement and the while() statement for easy readability.*

   d) Compare your script with this code. Make any adjustments if necessary.



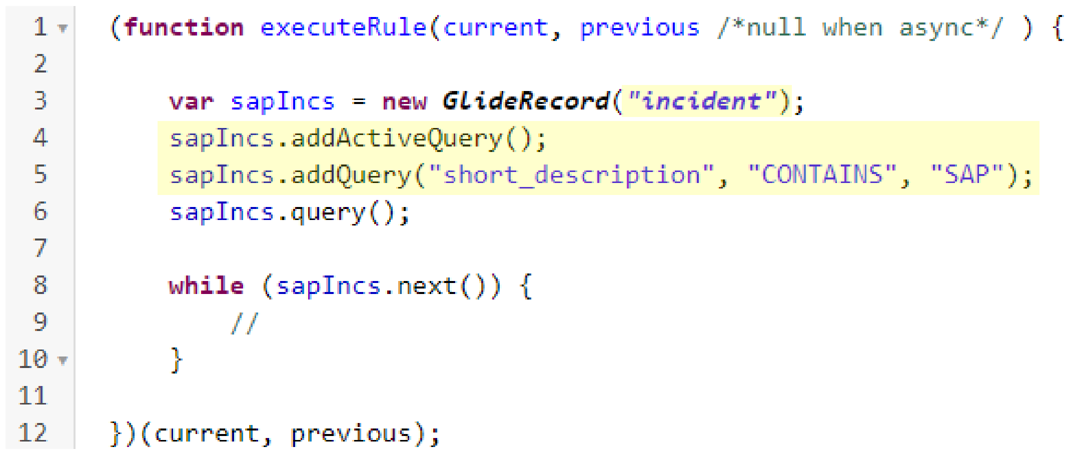
```
(function executeRule(current, previous /*null when async*/ ) {

    var sapIncs = new GlideRecord("");
    sapIncs.addQuery("name", "value");
    sapIncs.query();

    while (sapIncs.next()) {
        //
    }

})(current, previous);
```

6. Update the GlideRecord query to find all **active Incidents** where **Short Description contains SAP**.



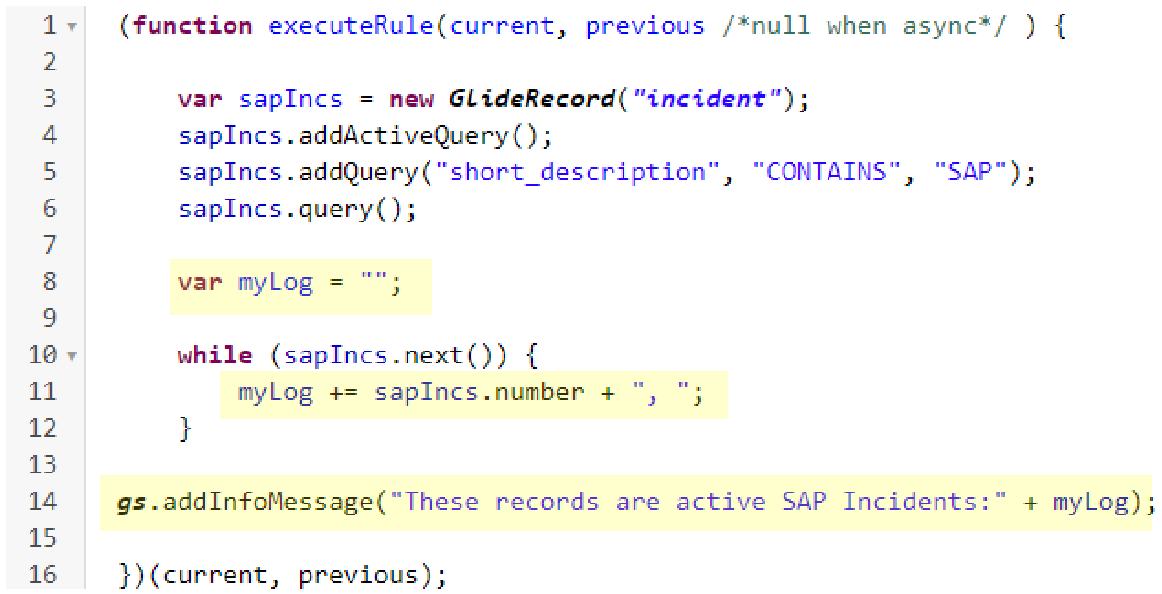
```
(function executeRule(current, previous /*null when async*/ ) {

    var sapIncs = new GlideRecord("incident");
    sapIncs.addActiveQuery();
    sapIncs.addQuery("short_description", "CONTAINS", "SAP");
    sapIncs.query();

    while (sapIncs.next()) {
        //
    }

})(current, previous);
```

7. Now update the Script to identify the Incidents matching the query criteria by their Number. You can use this script as a guide or any other identification strategy of your choice.



```
(function executeRule(current, previous /*null when async*/ ) {

    var sapIncs = new GlideRecord("incident");
    sapIncs.addActiveQuery();
    sapIncs.addQuery("short_description", "CONTAINS", "SAP");
    sapIncs.query();

    var myLog = "";

    while (sapIncs.next()) {
        myLog += sapIncs.number + ", ";
    }

gs.addInfoMessage("These records are active SAP Incidents:" + myLog);

})(current, previous);
```

8. Select **Submit**.

B. Test Your Work

1. Type **incident.list** in the Application Navigator's Search field and press **<Enter>** on your keyboard to open the list view of the Incident table.

2. Use the Condition Builder to build the same query you just scripted.

3. Run the query and record the number of returned rows here: _______________.

4. Create a new Incident and populate the mandatory fields with values of your choice.

5. **Save** the record to remain on the form.

6. Did your script locate the expected records? If not, debug and re-test your Business Rule.

7. Make the Lab 7.1 SAP Incidents Business Rule **inactive**.

C. Script 2: Update Executive User Records

1. Using the GlideRecord strategies, you have learned so far, create, test, and debug a GlideRecord query on your own.

   a) Find all users with Vice, VP or Chief in their title.

   b) Make the users VIPs.

   > **Hint:** *VIP is a Boolean field on the User [sys_user] table. The VIP field is not displayed on the User records unless the form/list is configured.*
   >
   > **Hint:** *Use the GlideRecord update() method to modify the User records.*

   > **Note***: If you are an experienced scripter, try creating your query in a Fix Script or Scheduled Job (without looking at the provided code, of course). However, if you are new to scripting and require assistance, an example Business Rule that can be used as a guide is on the next page.*

2. Did your script locate and update the expected records? If not, debug and re-test your Business Rule.

3. Make the Business Rule **inactive**.

Lab Completion

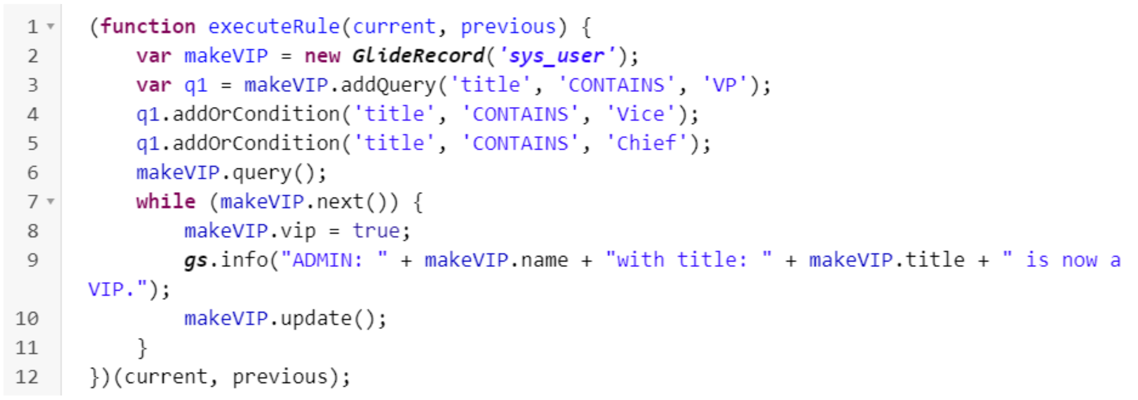
Great job querying the database using GlideRecord methods. If you were able to complete the second Business Rule on your own, bonus points for you. If not, don't feel bad! It took the sock monkeys months to figure it out.

D. Business Rule Example for Script 2

1. Create a new Business Rule.

   Name: **Lab 7.1 VIP Users**
   Table: **Incident [incident]**
   Active: **Selected (checked)**
   Advanced: **Selected (checked)**
   When: **after**
   Order: **100**
   Insert: **Selected (checked)**
   Update: **Selected (checked)**

   Script:



```
(function executeRule(current, previous) {
    var makeVIP = new GlideRecord('sys_user');
    var q1 = makeVIP.addQuery('title', 'CONTAINS', 'VP');
    q1.addOrCondition('title', 'CONTAINS', 'Vice');
    q1.addOrCondition('title', 'CONTAINS', 'Chief');
    makeVIP.query();
    while (makeVIP.next()) {
        makeVIP.vip = true;
        gs.info("ADMIN: " + makeVIP.name + "with title: " + makeVIP.title + " is now a VIP.");
        makeVIP.update();
    }
})(current, previous);
```

2. Test the Script

   a) Use the Condition Builder on the Users [sys_user] table to build the same query you just scripted. Make a mental note of the number of returned rows.

   b) Create a new Incident.

   c) **Submit** the record.

   d) Navigate to **System Logs > Script Log Statements** to review the log message.

   e) Did your script locate the expected records? If not, debug and re-test your Business Rule.

   f) Make the Business Rule **inactive**.