Gmail **Awad Bin-Jawed <awadbinjawed@gmail.com>**

GlideAggregate

1 message

Awad Bin-Jawed <awadbinjawed@gmail.com> Sun, Apr 14, 2019 at 9:51 AM
Draft To: Awad Bin-Jawed <awadbinjawed@gmail.com>

GlideAggregate

+ Subscribe ...

Jakarta

The `GlideAggregate` class is an extension of `GlideRecord` and allows database aggregation (COUNT, SUM, MIN, MAX, AVG) queries to be done. This can be helpful in creating customized reports or in calculations for calculated fields.

> **Note:** This functionality requires a knowledge of JavaScript.

For additional information, refer to GlideAggregate API.

See the JavaScript API for API information.

GlideAggregate examples

`GlideAggregate` is an extension of `GlideRecord` and its use is probably best shown through a series of examples.

> **Note:** This functionality requires a knowledge of JavaScript.

Here is an example that simply gets a count of the number of records in a table:

```
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if(count.next())
   incidents = count.getAggregate('COUNT');
```

There is no query associated with the preceding example. If you want to get a count of the incidents that were open, simply add a query as is done with `GlideRecord`. Here is an example to get a count of the number of active incidents.

```
var count = new GlideAggregate('incident');
count.addQuery('active','true');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if(count.next())
   incidents = count.getAggregate('COUNT');
```

To get a count of all the open incidents by category the code is:

```
var count = new GlideAggregate('incident');
count.addQuery('active','true');
count.addAggregate('COUNT','category');
count.query();
while(count.next()){
  var category = count.category;
  var categoryCount = count.getAggregate('COUNT','category');
  gs.log("there are currently "+ categoryCount +" incidents with
 a category of "+ category);}
```

The output is:

```
 *** Script: there are currently 1.0 incidents with a category of
Data
       *** Script: there are currently 11.0 incidents with a cate
gory of Enhancement
       *** Script: there are currently 1.0 incidents with a categ
ory of Implementation
       *** Script: there are currently 197.0 incidents with a cat
egory of inquiry
       *** Script: there are currently 13.0 incidents with a cate
gory of Issue
       *** Script: there are currently 1.0 incidents with a categ
ory of
       *** Script: there are currently 47.0 incidents with a cate
gory of request
```

The following is an example that uses multiple aggregations to see how many times records have been modified using the *MIN*, *MAX*, and *AVG* values.

```
var count = new GlideAggregate('incident');
count.addAggregate('MIN','sys_mod_count');
count.addAggregate('MAX','sys_mod_count');
count.addAggregate('AVG','sys_mod_count');
count.groupBy('category');
count.query();
while(count.next()){
  var min = count.getAggregate('MIN','sys_mod_count');
  var max = count.getAggregate('MAX','sys_mod_count');
  var avg = count.getAggregate('AVG','sys_mod_count');
  var category = count.category.getDisplayValue();
  gs.log(category +" Update counts: MIN = "+ min +" MAX = "+ max
+" AVG = "+ avg);}
```

The output is:

```
       *** Script: Data Import Update counts: MIN = 4.0 MAX = 21.
0 AVG = 9.3333
       *** Script: Enhancement Update counts: MIN = 1.0 MAX = 44.
0 AVG = 9.6711
       *** Script: Implementation Update counts: MIN = 4.0 MAX =
8.0 AVG = 6.0
       *** Script: inquiry Update counts: MIN = 0.0 MAX = 60.0 AV
G = 5.9715
       *** Script: Inquiry / Help Update counts: MIN = 1.0 MAX =
3.0 AVG = 2.0
       *** Script: Issue Update counts: MIN = 0.0 MAX = 63.0 AVG
= 14.9459
       *** Script: Monitor Update counts: MIN = 0.0 MAX = 63.0 AV
G = 3.6561
       *** Script: request Update counts: MIN = 0.0 MAX = 53.0 AV
G = 5.0987
```

The following is a more complex example that shows how to compare activity from one month to the next.

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count','category');
agg.orderBy('category');
agg.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(2)');
agg.query();
while(agg.next()){
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count','category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last ="";
  while(agg2.next()){
     last = agg2.getAggregate('count','category');}
  gs.log(category +": Last month:"+ count +" Previous Month:"+ last);

}
```

The output is:

```
*** Script: Monitor: Last month:6866.0 Previous Month:4468.0
*** Script: inquiry: Last month:142.0 Previous Month:177.0
*** Script: request: Last month:105.0 Previous Month:26.0
*** Script: Issue: Last month:8.0 Previous Month:7.0
*** Script: Enhancement: Last month:5.0 Previous Month:5.0
*** Script: Implementation: Last month:1.0 Previous Month:0
```

The following is an example to obtain distinct count of a field on a group query.

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count');
agg.addAggregate('count(distinct','category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
//
agg.groupBy('priority');
agg.query();
while (agg.next()) {
// Expected count of incidents and count of categories within each priority va
lue (group)
  gs.info('Incidents in priority ' + agg.priority + ' = ' + agg.getAggregate(
'count') +
            ' (' + agg.getAggregate('count(distinct','category') + ' categorie
s)');
}
```

The output is:

```
*** Script: Incidents in priority 1 = 13 (3 categories)
*** Script: Incidents in priority 2 = 10 (5 categories)
*** Script: Incidents in priority 3 = 5 (3 categories)
*** Script: Incidents in priority 4 = 22 (6 categories)
```

You can implement the SUM aggregate with or without the use of the `groupBy()` method. If you do not use the `groupBy()` method, the result of the SUM is the cumulative value for each different value of the field for which you request the SUM. For example, if you SUM the total_cost field in the Fixed Asset table, and the Fixed Asset table contains 12 total records:

- Three records with a total_cost of $12
- Four records with a total_cost of $10
- Five records with a total_cost of $5

When you SUM the record set, the `getAggregate()` method returns three different sums: $36, $40, and $25.

The following code illustrates implementing the SUM aggregate without using the `groupBy()` method:

```
var totalCostSum = new GlideAggregate('fixed_asset');
totalCostSum.addAggregate('SUM', 'total_cost');
totalCostSum.query();

while (totalCostSum.next()) {
  var allTotalCost = 0;
  allTotalCost = totalCostSum.getAggregate('SUM', 'total_cost');
  aTotalCost = totalCostSum.getValue('total_cost');
  gs.print('Unique field value: ' + aTotalCost + ', SUM = ' + allTotalCost +
', ' + allTotalCost/aTotalCost + ' records');
}
```

The output for this example is:

```
*** Script: Unique field value: 12, SUM = 36, 3 records
*** Script: Unique field value: 10, SUM = 40, 4 records
*** Script: Unique field value: 5, SUM = 25, 5 records
```

Using the same data points as the prior example, if you use the `groupBy()` method, the SUM aggregate returns the sum of all values for the specified field.

The following example illustrates implementing the SUM aggregate using the `groupBy()` method:

```
var totalCostSum = new GlideAggregate('fixed_asset');
totalCostSum.addAggregate('SUM', 'total_cost');
totalCostSum.groupBy('total_cost');
totalCostSum.query();
if(totalCostSum.next()){  // in case there is no result
   var allTotalCost = 0;
   allTotalCost = totalCostSum.getAggregate('SUM', 'total_cost');
   gs.print('SUM of total_cost: = ' + allTotalCost);
}
```

The output for this example is:

```
*** Script: SUM of total_cost: 101
```