SK Reporting Engine

Developed by Prospect BD

SK Reporting Engine has Two part Scripting , Reporting

- Scripting
- In Scripting part you can declare Variables and do relational Data Process.
- Variable Declaration
- Variable name should be Start with ‘$’, variable declarations should be separated with ‘;’,
- Examples:
- $x = 10;
- $y = 100;
-
- Variable types are Integer , Float, String, Json, Array and Dictionary (Key value Pair).
- examples:
- $x = 10;   \ integer
- $y = 10.10;   \float
- $z = ‘gkibria’;   \string
- $a = [1,2,3,4,5,6] \array or list
- $b = {‘key1’ : ‘value1’, ‘key2’ : ‘value2’ ’, ‘key3’ : ‘value3’ ’, ‘key4’ : ‘value4’}; \dictionary

- $c = {
-   "person": {
-     "name": "Alice",
-     "age": 28,
-     "location": "New York",
-     "email": "alice@example.com"
-   },
-   "favorite_fruits": ["apple", "banana", "orange"],
-   "is_student": false,
-   "grades": {
-     "math": 92,
-     "science": 85,
-     "history": 78 },
-   "addresses": [{"type": "home","street": "123 Main St","city": "Springfield","country": "USA"},
-     {
-       "type": "work",
-       "street": "456 Elm St",
-       "city": "New York",
-       "country": "USA"
-     }
-   ]
- };   \json

- Single-line comments with '/', and multi-line comments with '/* */':plaintext// This is a single-line comment/* This is   a multi-line   comment */

- **Data Process**
- **$new_table_name = $<table1:placeholder1 , table2:placeholder2 >operation1(arguments1)**
- **-> operation2(arguments2) -> operation3(arguments3)……..…;**
- You can perform **SELECT, WHERE , GROUP, SORT, JOIN**
- Examples:
- **If you want to Join two tables.**
- **$new_table = $<table1:p1,table2:p2>join(attribute_name:p1.local_id=p2.foreign_id);**

- Example:
- **$users_posts = $<users:x, posts:y>join(posts:x.id=y.userId);**
- The join type is left joint
- If tables has one to many relationship use '_' after attribute_name
- Example :
- **$new_table = $<table1:p1,table2:p2>join(attribute_name_:p1.local_id=p2.foreign_id);**

- **To SELECT**
- $new_table = $<table1:x>select({‘key1’ : ‘value1’ ,’key2’ : x.key1+x.key2+(x.key3/x.key4) });
- Supported Operations :
- + / add two fields
- - / subtract two fields
- / / divide two fields
- * /multiply two fields
- () /give precedence to operators
- Function(expression) / evaluate function exmpale : sum,avg,sing,cos

- Example :
- $new_table =

- **To filter the table you can use WHERE**

- **$new_table = $<table:x>where(x.id>1)->select({ ‘id’ : x.id   });**

- Supported operators :

>	/greater then
>=	/greater then equal
== or =	/equal to
<	/less then
<=	/less then equal
!=	/not equalt
()	/to nested logic

- Conditional operator
- AND
- OR
- NOT

- Example:
- $new_table = $<table:x>where(x.id > 1 AND x.name ==‘gkibria’);

- **To Sort Table**
- $new_table = $<table:x>sort(fieldname:asc|des, fieldname:asc|des);
- Example : $table2 = $<table:x>sort(x.item,x.name:asc,x.age:des);

- **To group table**

- $new_table = $<table:x>group(fieldname, fieldname,fieldname)->select({‘key1’ : ‘value1’});
- Example:
- $new_table = $<table:x>group(x.age)->select({ ‘id’ : sum(id) , ’first_name’ : x.name, ‘age’: x.age   })

- Note : You can use function in the select

- **Expression in Variable**
- If you want to evaluate any expression in script the syntax is

- $varaiable_name = @’expression’;

- Example : $x = @’1+2+sin(90)’; outupt : $x = 4;

- If you don’t want to evaluate expression :
- $x = ‘ “expression” ‘;
- Example : $x = ’ “1+2+sin(90) “ ‘; output : $x = “1+2+sin(90)”;
-

Reporting

Reporting has four part, import Template , Perform function, perform format and execute Script

To print Variable : {{$variable}}
Example :
This is a variable {{$x}}

To import Template from an other folder
Syntax = {{::file_path,   placeholder_values_as_dectionary   }}
The file should be in the template directory , it can in be any level of sub folders
Supported files ar .txt and .tmpl

Examples:

```
{{::template1}}   // the template1.txt file is in template folder
{{::sub/template2}}  // the template2.tmpl is in template/sub/ folder
```

- Suppose template.txt file has
- This is the {{$user_placeholder}}

- To replace the placeholde
- {{::template, {‘$user_placeholder’ : ‘$user’}}} // here $user is a variable stored in the data structure

Nested Level json

If the variable is a list access it by [index] if it’s a object access by .attribute

Exmpale :

$x = [1,2,3,4,5,6,[1]];

In template :

{{$x[6][0]}} output : 1

$x = {‘a’ : {‘b’ : ‘c’}}

In template :

{{$x.a.b}} output: c

if the variable is an json list then access element by brace convention if it’s a object then access it by dot convention

Example :

$x = [{‘a’ : [1,2,3,4]}];

In template :

{{$x[0].a[1]}} output : 2

Template Format

You can format a variable by two way

1 . In line

2. Using Format classes(you have to define the format class in the template using <format></format>)

In line format :

{{$variable_name::format_sepc_as_an_object}}

Example :{{$x::{‘width’ : 5, ‘align’ : ‘center’ , ‘fill’ : ‘$’}}} /$x = 1;

Output : $$1$$

When you use class format you can use conditional format

Example :
{{$x:((y)=> y>1), c1|c2}}

Here ((y)=> y>1) is the condition , if it’s true then the format will be c1 else c2

Exmple :
```
{{$x:((y)=> y>1), c1|c2}}      \ $x = 1
<format>
c1 = {‘width’ : 4 , ‘aling’ : ‘left’ ,’fill’ : ‘$’}
c2 = {‘width’ : 4 , ‘aling’ : ‘right’,’fill’ : ‘$’}
</format>
```

Output : 1$$$
If $x = 2
Output : $$$1

- You can use Aggregate methods in variable
- Example :
- {{$x.sum()}} \\ $x = [1,2,3,4] output : 10

- Functions are sum,avg,camel,capitalize,ceil etc

- Foreach method:

- Syntax :
- {{$variable_list.foreach(($element)=>{   <sub template>   }   )}}

- Exmaple:
- In script
- $x = [{'key' : 'value1'} , {'key' : 'value2'} , {'key' : 'value3'}];

- In template:
- {{$x.foreach(($y)=>{   {{$y.key}}   })}}   output: value1   value2   value3