Glide Record

- It is a server-side API
- It is used for database Operation.
- The Glide Record API is a primary means of interfacing with the database on the server-side code.
- A Glide Record is a Object that contain Record from single table [EX= incident , problem , change etc].
- Use this API to represent a Glide Record & add Query parameter, filter, Limit, Ordering etc…

Glide Record Exercises

1. How to get **result(output)** in ServiceNow

- gs.print ('Welcome to ServiceNow Academy');
- gs.info ('Welcome to ServiceNow Academy');

- **Result:=** Welcome to ServiceNow Academy

Write a simple program add two numbers

- var a = 10;
- var b = 20;
- var c = a+b;
- gs.print (c);

- **Result:= 30**

Working with query() method

- var inc = new GlideRecord ('incident’) //GlideRecord is main Oject and Incident is Table
- inc.query (); //Query is execute in the table
- while (inc.next ()) { //Loop will runs on the table
- gs.print (inc.number); //Printing all incidets
- }

- **Result:-** Print all records numbers in **Incident Table**

Exercise -1: Display **priority -1** tickets from incident table with **addQuery** methods

- var inc = new GlideRecord ('incident');
- inc.addQuery ('priority=1');// Add the query
- inc.query ();
- while(inc.next()){
- gs.print(inc.number);
- }

●

- **Result**:-Printing all **prority-1 tickets**

Exercise-2: Passing **Multiple Queries** using by same methods

- var inc = new GlideRecord('incident');
- inc.addQuery ('active', true);   //Query 1
- inc.addQuery ('priority=1');   //Query 2
- inc.addQuery ('category','software');   //Query 3
- inc.query ();
- while(inc.next()){
- gs.print (inc.number);
- }

- **Result:-** Print all records where your **Condition meet**

Exercise-3: we can use **addEncodedQuery** method Instead of passing multiple queries into our script

- **Step-1:** Navigate to Incident **list view** and apply condition
- **Step-2: Condition:** active = true and priority =1 and category = software
- **Step-3:** Click on **Run**
- **Step-4: Copy** applied query through **Copy query**
- **Step-5:** Use this entire query into your script
- **Step-6:** Script
- var inc = new GlideRecord ('incident');
- inc.addEncodedQuery('active=true^category=software^priority=1');
- inc.query();
- while(inc.next()){
- gs.print(inc.number);
- }

Exercise-4: Encoded Query set to a variable that variable to call into code.

- var ecq = 'active=true^category=software^priority=1’; //encodedquery set to a variable
- var inc = new GlideRecord('incident');
- inc.addEncodedQuery (ecq);
- inc.query();
- while (inc.next()){
- gs.print (inc.number);
- }

- **Result:-**Print all records where this meet **'active=true^category=software^priority=1';**

Working with **addQuery** (‘String’,’Operator’,’Value’)

- =
- !=
- >
- >=
- <
- <=
- **Strings (must be in upper case):**
- =
- !=
- IN
- NOT IN
- STARTSWITH
- ENDSWITH
- CONTAINS
- DOES NOT CONTAIN
- INSTANCEOF

Exercise-5: Get **Active** and **Priority is less than** or **equal to 2**

- var inc = new GlideRecord('incident');
- inc.addActiveQuery();
- inc.addQuery('priority','<=',2);
- inc.query();
- while(inc.next()){
-   gs.print(inc.number);
- }

- **Result:-**Print **Critical-1** and **High-2** tickets

Exercise-7: Working with **SQL operators <=** and **CONTAINS**

- var inc = new GlideRecord('incident');
- inc.addActiveQuery();
- inc.addQuery('priority','<=',2);
- inc.addQuery('short_description','CONTAINS’,’test’);
- inc.query();
- while(inc.next()){
- gs.print(inc.number + ' ' + inc.short_description);
- }

- **Result:-**Print all records where our condition meet like **(<=2 and CONTAINS)**

Exercise-8: Working with **IN** operator and print category of **Software** and **Hardware**

- var cat = ['software', 'hardware'];
- var inc = new GlideRecord('incident');
- inc.addQuery('category', 'IN', cat);
- inc.query();
- while(inc.next()) {
- gs.print(inc.getValue('number') + ' ' + inc.getValue('short_description')) ;
- }
- **Result:-**Print where category is **Software ad Hardware**

Exercise-9: Working with **STARTSWITH** Operator

- var inc = new GlideRecord('incident');
- inc.addQuery('category', 'STARTSWITH', 'net');
- inc.query();
- while(inc.next()) {
- gs.print(inc.number);
- }
- **Result:-**Print where category startswith net.

Exercise-10: Instead of use **active=true** this method directly we can use **addActiveQuery**

- var inc = new GlideRecord('incident');
- inc.addActiveQuery ();// instead if passing active = true
- inc.addQuery ('priority',1);
- inc.query ();
- while (inc.next ()){
-   gs.info (inc.number);
- }
- **Result:-** Print all records where condition is equal to **active** is **true** and **priority-1**

Exercise-10: Instead of use **active=false** this method directly we can use **addInactiveQuery**

- var inc = new GlideRecord ('incident');
- inc.addInactiveQuery (); //Opposite of active query
- inc.addQuery ('priority=1');
- inc.query ();
- while (inc.next ()) {
-   gs.print (inc.number);
- }

- **Result:-** Print only inactive Records like Incident state is **Closed**

Working with **orderBy ()** method

Exercise-12: Display all records in order wise **(Assending)** it depends on field values.

- var inc = new GlideRecord('incident');
- inc.addQuery('priority=1');
- inc.addQuery('category=software');
- inc.orderBy('short_description');
- inc.query();
- while(inc.next()){
- gs.print(inc.number + ' ' + inc.short_description);
- }
- **Result:-**Print all incidents order wise depends on **Short Description**

Working with **orderByDesc ()** method

Exercise-13 Display all records in order wise **(Descending)** it depends on field values

- var inc = new GlideRecord('incident');
- inc.addQuery('priority=1');
- inc.addQuery('category=software');
- inc.orderByDesc('short_description');
- inc.query();
- while(inc.next()){
- gs.print(inc.number + ' ' + inc.short_description);
- }
- **Result:-**Print all records in descending order **(short_description)**

Working with **setLimit ()** method

Exercise-14: Display limited records from specified table

- var inc = new GlideRecord('incident');
- inc.addQuery('priority=1');
- inc.orderByDesc('short_description');
- inc.setLimit(10);
- inc.query();
- while(inc.next()){
- gs.print(inc.number + ' ' + inc.short_description);
- }

- **Result:-** Print only latest **10 records** created from given table

Working with **get ()** Method

Exercise-15: Get record **sys_id** depends on **INC number** or Get incident record number depends on **sys_id**

- var inc = new GlideRecord('incident');
- inc.get('number','INC0009005');
- gs.print(inc.sys_id);

- **Result:-**Print sys_id related to **incident number**

Working with **getRowCount ()** method

Exercise-16: Display all records from particular table (Incident)

- var inc = new GlideRecord('incident');
- inc.query()
- gs.print(inc.getRowCount());

- **Result:-**Print number of records in particular table

Working getTableName () method

Exercise-17: This method is used to get glide record table name

- var inc = new GlideRecord ('change_request');
- gs.print (inc.getTableName ());
-
- **Result:-** Display current table name from glide record.

Working getValue () method

Exercise-18: Get value of particular field in the table

- var inc = new GlideRecord('incident');
- inc.addQuery('active=true');
- inc.query();
- while(inc.next()){
-   gs.print(inc.getValue('short_description'));
- }
-
- **Result:-** Print the value of field from particular table

Working getDisplayValue () method

Exercise-19 Print display value instead of actual value.

-
- var inc = new GlideRecord('incident');
- inc.addQuery ('priority=1')
- inc.query ();
- while (inc.next ()){
- gs.print (inc.priority.getDisplayValue ());
- }
-
- **Result:-**Print display value of respective field

Working hasNext () method

Exercise-20: This method will return true if iterator have more elements.

- var inc = new GlideRecord ('incident');
- inc.query ();
- gs.print (inc.hasNext ());
-
- **Result:-** Print Boolean value **(True/False)**

Working with getUniqueValue () method

Exercise-21: Gets the uniue key of the record, which is usually the sys_id unless otherwise specified.

- var inc = new GlideRecord('incident');
- inc.query();
- inc.next();
- var uniqvalue = inc.getUniqueValue();
- gs.print(uniqvalue);

- **Result:-** Sys_id of incident table

Working with **setValue ()** method

Exercise-22: This method is used to sets the value of the specific field with the specified value.

- var fieldName = 'category';
- var inc = new GlideRecord ('incident');
- inc.initialize ();
- inc.setValue(attriName,'network');
- inc.setValue('short_description’,’Outlook issue');
- inc.insert();
- gs.print ('Category is ' + inc.category + ' and ' + 'issue is: ' + inc.short_description);
-
- **Result:-**Create a new record and **Set** a value into **category** field and **short description.**

Working with **initialize ()** and **insert ()** method

Exercise-23: These methods are used to **Inserts a new record** using the field values that have been set for the current record

•

var inc = new GlideRecord ('incident');
- inc.initialize ();   //Compose incident form
- inc.category = ‘network’;   // set field values
- inc.short_description = ‘Firewall Issue’;
- inc.priorty = 1;
- inc.insert ();   // create new record
- gs.print (inc.number);// print new record incident number

- **Result:-** Create new record and print new record number

Working with **isNewRecord ()** and **newRecord ()** method

Exercise-24: Checks if the current record is a new record that has not yet been inserted into the database.

-
- var inc = new GlideRecord ('incident');
- inc.newRecord ();
- gs.info (inc.isNewRecord());

Result:- Return bool value true or false **(value is True)**

Working with **addNullQuery ()** method

Exercise-25: display all records where the value of the specified field is null.

- var inc = new GlideRecord('incident');
- inc.addNullQuery ('short_description')
- inc.query ();
- while (inc.next ()) {
- gs.print (inc.number)

}

Result:- Print all records where the specific field value is **Null**

Working with **addNotNullQuery ()** method

Exercise-26: Opposite of addNullQuery methods display all records where the value of the specified field is **not null.**

- var inc = new GlideRecord('incident');
- inc.addNotNullQuery ('short_description')
- inc.query ();
- while (inc.next ()) {
- gs.print (inc.number)
- }
-
- **Result:-**Print all records where the specific field value is **not null**

Working with **update ()** method single record

Exercise-27: Update specific record from table

- var inc = new GlideRecord ('incident');
- inc.get ('number','INC0000057');
- inc.setValue ('state', 2);
- inc.update ();

- **Result:-** update record as expected

Working with **updateMultiple ()** method multiple record

Exercise-28: Updates multiple records in a stated query with a specified set of changes from respected table**.**

- var inc = new GlideRecord('incident');
- inc.addQuery ('category', 'hardware');
- inc.setValue('category', 'software');
- inc.updateMultiple(); ();

- **Result:-** Update multiple records as expected

Working with **deleteMultiple ()** method multiple record

Exercise-29: Deletes multiple records that satisfy the query condition.

- var inc = new GlideRecord('incident');
- inc.addQuery('priority', 4);
- inc.query ();
- inc.deleteMultiple ();

- **Result:-** Delete multiple records as expected