

ArrayUtil API

Merges two arrays	<code>new ArrayUtil().concat(array1, array2)</code>
Searches the array for the element. Returns true if the element exists in the array, otherwise returns false	<code>new ArrayUtil().contains(array, search object)</code>
Converts an object to an array.	<code>new ArrayUtil().convertArray(Object)</code>
Finds the differences between two or more arrays	<code>new ArrayUtil().diff(array1, array2, array n)</code>
Returns an array from the object.	<code>new ArrayUtil().ensureArray(Object)</code>
Searches the array for the element. Returns the element index if found, -1 otherwise.	<code>new ArrayUtil().indexOf(array1, search element)</code>
Finds the elements common in all arrays.	<code>new ArrayUtil().intersect(Array a, Array b)</code>

GlideFilter

Compares a specified filter to the contents of a specified GlideRecord.	<code>GlideFilter.checkRecord(glideRecord, attribute);</code>
---	---

If the specified filter contains one condition, the method returns true if the record meets the condition of the filter.

matchparameter indicates whether all conditions must be met if the filter parameter contains multiple conditions.

Valid values:

true: all conditions must be met for the method to return true

false: only one of the conditions must be met for the method to return true.

TableUtils

Checks to see if a table exists.	<code>new TableUtils("TABLENAME").tableExists()</code>
Drops a database table. (use with caution)	<code>new TableUtils().drop("TABLENAME")</code>
Drops a database table and cleans up references to the table.	<code>new TableUtils().dropAndClean("TABLENAME")</code>
Drops a database table, all of it's extended tables, and cleans up references to the tables.	<code>new TableUtils().dropTableAndExtendedTables("TABLENAME");</code>

Merges two or more arrays (common element is added once)

```
new ArrayUtil().union(Array a, Array b)
```

Removes duplicate items from an array.

```
new ArrayUtil().unique(Array a)
```

The syntax of the above APIs can be Like this

```
var arUtil = new ArrayUtil();  
arUtil.union(array1);
```

CIUtils

Determine which business services are affected by a specific CI.

```
new CIUtils().servicesAffectedByCI(ci_sys_id);
```

Determine which business services are affected by a task.

```
new CIUtils().servicesAffectedByTask(task_id)
```

Sample Usage of the script can be like below

```
var CIUtil = new CIUtils();  
var bsaffected = CIUtil.servicesAffectedByCI(ci_sys_id);
```

Returns the base table name from which the table was extended.

```
new TableUtil(s("table_name")).getBaseTable();
```

Returns the list of tables that extend a table, includes the base table in array

```
new TableUtil(s("table_name")).getAllExtends();
```

Returns a list of all classes participating in the hierarchy of the specified table.

```
new TableUtil(s("table_name")).getHierarchy();
```

Returns the table hierarchy.

```
new TableUtil(s("table_name")).getTableHierarchy();
```

Returns a list of tables that extend a table.

```
new TableUtil(s("table_name")).getTables();
```

Determines if a table has been extended.

```
new TableUtil(s("table_name")).hasExtended();
```



By **Bibin Gokuldas**
(bibingokuldas)

Published 24th August, 2020.
Last updated 24th August, 2020.
Page 1 of 11.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

cheatography.com/bibingokuldas/
bibingokuldas.com/

TableUtils (cont)

Determines `new TableUtils("tablename").isBaseClass()`
if a table is `class();`
 a base
 class,
 meaning it
 has no
 parents
 and has
 extensions.

Determines `new TableUtils("tablename").isSoloclass();`
if the table `class();`
 has no
 parents
 and no
 extensions.

Use Drop the table function with extreme caution as it will delete the table and the data forever

GlideSysAttachment

SLARepair

Repair the `new SLARepair().repairByFilter('filtername');`
 task SLAs
 associated
 with the
 passed-in
 filter and
 source
 table.

Repair the `new SLARepair().repairByGlideRecord()`
 task SLAs
 associated
 with the
 passed in
 GlideR-
 ecord.

Repair the `new SLARepair().repairBySysId('sys_id');`
 task SLAs
 associated
 with the
 passed in
 sys_id and
 source
 table.

Enables or `new SLARepair().setAuditEnabled(true);`
 disables
 auditing
 when
 running a
 repair.*

Enables or `new SLARepair().setRunWorkflow(false);`
 disables
 running a
 workflow
 for each of
 the Task
 SLA
 records
 being
 repaired.

Validates `new SLARepair().setValidateOnly(true);`
 the repair
 request.

The SLARepair API first deletes the existing SLAs and then recreates them from each task's history.
**auditing is set to the value in the property com.snc.sla.repair.audit.*
You can override this with passing in true to enable or false to disable auditing.

GlideEmailOutbound

Copies attachments from the source record to the target record.	<code>new GlideSysAttachment().copy(String tableName, String emailID, String targetEmail, String address2');</code>	Adds the email address to either the cc or bcc list.
Deletes the specified attachment.	<code>new GlideSysAttachment().deleteAttachment(Integer.parseInt(sysID))</code>	Returns the email's subject line.
Returns a GlideRecord containing the matching attachment metadata such as name, type, or size.	<code>new GlideSysAttachment().getAttachments(Integer.parseInt(sysID))</code>	Returns the email's watermark.
Returns the attachment content as a string.	<code>new GlideSysAttachment().getBody(String sysID)</code>	Sets the body of the email.
Returns the attachment content as a string with base64 encoding.	<code>new GlideSysAttachment().getContent(String sysID)</code>	Sets the sender's address.
Inserts an attachment for the specified record using base64 encoded content.	<code>new GlideSysAttachment().writeBase64(GlideRecord sysAttachment, String fileName, String contentType, String content)</code>	Sets the reply to address.
		Sets the email's subject line.



By **Bibin Gokuldas**
(bibingokuldas)

Published 24th August, 2020.
Last updated 24th August, 2020.
Page 2 of 11.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

GlideSession

Sets a session client value that can be retrieved with getClientData().

```
gs.getSession().putClientData('parameter name', 'parameter value');
```

Determines if the current user is currently logged in.

```
'gs.getSession().isLoggedIn()'
```

Determines if the current session is interactive.

```
'gs.getSession().isInteractive()'
```

Gets the name of the session's time zone.

```
'gs.getSession().getTimeZoneName()'
```

Gets a list of roles for the current user.

```
'gs.getSession().getRoles()'
```

Gets the session's language code.

```
'gs.getSession().getLanguage()'
```

Returns a session client value previously set with putClientData().

```
gs.getSession().getClientData('parameter name');
```

Clears a session client value previously set with putClientData().

```
gs.getSession().clearClientData('parameter name');
```

The GlideSession API allows you to find information about the current session.

```
var session = gs.getSession();
session.putClientData('cheat sheet', 'OOBAP I');
```

CartJS (cont)

Returns the GlideRecord for the cart item (sc_cart_item) in the current cart.

```
new sn_sc.CartJS().getCartItem();
```

Gets the sys_id from the sys_user record of the user for whom the cart is requested.

```
new sn_sc.CartJS().getRequestedFor();
```

Orders a single item.

```
new sn_sc.CartJS().orderNow(request);
```

Specifies if the current user has the required role to edit the Request for field.

```
new sn_sc.CartJS().canViewRF();
```

Sets the sys_id in the sys_user record of the user for whom the cart is requested.

```
new sn_sc.CartJS().setRequestedFor(sysid)
```

The variable used in the scripts above called 'request' should be a JSON Object which has the details of the catalog items to be added to cart or to order. Sample is below

```
var item = {
  'sysparm_id': '0d088 372 371 530 001 58b bfc 8bc - be5 d02', // Catalog item sys_id
  'sysparm_quantity': '1',
  'variables': { // Pass the catalog item variables
    'carrier': 'at_and_t_mobility',
```

CartJS

Adds the request for a catalog item to the current cart.

```
new sn_sc.CartJS().addToCart(request);
```

Performs the cart checkout. If the two-step checkout is enabled, returns the order summary. If the two-step checkout is disabled, the cart is submitted and details of the generated request are returned.

```
new sn_sc.CartJS().checkoutCart();
```

Deletes the current cart.

```
new sn_sc.CartJS().empty();
```

Returns the cart id of the current cart.

```
new sn_sc.CartJS().getCartID();
```

```
'data_plan': '500MB',  
'duration': 'eighteen_months',  
'color': 'slate',  
'storage': 'sixtyfour'  
}};
```

TemplatePrinter (Mail Script)

Prints the string to the email body.

```
template.print("Incident number - " + c  
" \n");
```

Adds non-breaking spaces to the email body.

```
template.space(4);
```

GlideLocale

Returns the GlideLocale object.

```
GlideLocale.get()
```

Returns the decimal separator.

```
var locale = GlideLocale.get(); var dsep  
rator()
```



By **Bibin Gokuldas**
(bibingokuldas)

cheatography.com/bibingokuldas/
bibingokuldas.com/

Published 24th August, 2020.
Last updated 24th August, 2020.
Page 3 of 11.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

GlideLocale (cont)

Returns the grouping separator.

```
var locale = GlideLocale.get();
var groupSeparator = locale.getGroupSeparator();
```

GlideRecordUtil

Returns a GlideRecord instance positioned to the given CI sys_id, and of the right class (table).

```
new GlideRecordUtil().getCIRecord("cisys_id", "sys_id", "table");
```

Returns a list of all the fields in the specified GlideRecord.

```
new GlideRecordUtil().getFields(glideRecordObject);
```

Returns a GlideRecord instance for the given table, positioned to the given sys_id, and of the right class (table).

```
new GlideRecordUtil().getRecord("table", "sys_id", "sys_id");
```

Returns a Java ArrayList of the ancestors of the given table name.

```
new GlideRecordUtil().getTables("tablename");
```

GlideSchedule

Adds a new schedule segment to the current schedule.

```
new GlideSchedule().addSegment(startDate, endDate, timeZone, duration);
```

Determines the elapsed time in the schedule between two date time values using the timezone of the schedule or, if that is not specified, the timezone of the session.

```
new GlideSchedule().getElapsedTime(startDate, endDate, timeZone);
```

Retrieves the schedule name.

```
new GlideSchedule(sys_id).getName();
```

Determines if the given datetime is within the current schedule.

```
new GlideSchedule("schedule_id").isWithin(startDate, endDate);
```

Determines if the current schedule is valid. A schedule is valid if it has at least one schedule span.

```
new GlideSchedule("schedule_id").isValid();
```

Sets the timezone for the current schedule.

```
new GlideSchedule().setTimeZone(timeZone);
```

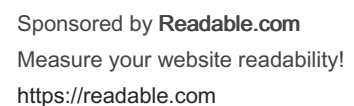
```
"Determines how much time (in millis-
econds) until start time of the
next schedule item."
```

FlowAPI

paused
or
running
flow,
subflow,
or
action.

```
Run an action from a server-side script synchronously.

sn_fd.FlowAP I.execute Action ('flow n
, inputs);
```



FlowAPI (cont)

Run an action from a server-side script synchronously from the current user session without creating execution details or other related records

```
sn_fd.FlowAPI.executeActionQuietly('flowname', sn_inputs);
```

Runs a Data Stream action synchronously from a server-side script and returns a ScriptableDataStream object.

```
sn_fd.FlowAPI.executeDataStreamAction('datastreamname', sn_inputs, timeout);
```

Run a flow from a server-side script synchronously.

```
sn_fd.FlowAPI.executeFlow('flowname', sn_inputs, timeout);
```

FlowAPI (cont)

Run a flow from a server-side script.

```
sn_fd.FlowAPI.executeFlow('flowname', sn_inputs);
```

Run a flow from a server-side script asynchronously without creating execution details

```
sn_fd.FlowAPI.executeFlowQuietly('flowname', sn_inputs);
```

Run a subflow from a server-side script.

```
sn_fd.FlowAPI.executeSubflow('subflowname', sn_inputs);
```

Run a subflow from a server-side script asynchronously without creating execution details

```
sn_fd.FlowAPI.executeSubflowQuietly('subflowname', sn_inputs, timeout);
```

Use FlowAPI methods to execute actions, flows, or subflows in server-side scripts using either blocking or non-blocking methods.

Run a `sn_fd.F lo wAP I.e xec ute Flo wQu ick ('f low name', inputs);`
flow,
subflow, or
action
from a
server-
side script
synchr-
onously or
asynch-
ronously
without
creating
execution
details or
other
related
records.

Run an `var results = sn_fd.F lo wAP I.e xec ute Sub flo w(' flo wname', inputs`
subflow `);`
from a
server-
side script
synchr-
onously.

Run a `var results = sn_fd.F lo wAP I.e xec ute Sub flo wQu ick ('f low name',`
subflow `inputs);`
from a
server-
side script
synchr-
onously
from the
current
user
session
without
creating
execution
details or
other
related
records

Build `var encryp ted dat a=s n_f d.G lid eAc tio nUt il.s et Enc ryp ted Out -`
password2 `put (Val);`
values
inside a
script step.

Run an `sn_fd.F lo wAP I.s tar tAc tio n(' flo wname', inputs);`
action
from a
server-
side script
asynch-
ronously.

Run an `sn_fd.F lo wAP I.s tar tAc tio nQu ick ('f low name', inputs);`
action
from a
server-
side script
asynch-
ronously
without
creating
execution
details



By **Bibin Gokuldas**
(bibingokuldas)

cheatography.com/bibingokuldas/
bibingokuldas.com/

Published 24th August, 2020.
Last updated 24th August, 2020.
Page 5 of 11.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>