SAP Gateway Usage with Any Programming Language

August 2020

RIZING

Introduction

With Fiori becoming the primary User Interface for SAP S/4HANA implementations, SAP Gateway is more important than ever. For many implementations of S/4HANA, the Fiori launchpad is the only way a business user will access SAP and will never use the classical SAPGUI. Fiori applications will utilize an SAP Gateway RESTful web service to communicate with the SAP backend system. For ABAP developers, this is a very critical skill to have for S/4HANA development and these skills could complement future development with SAP Cloud Platform OData Provisioning.

SAP has embraced the "Bring Your Own Language" approach to developing applications with SAP. With SAP Gateway, creating RESTful web services allows for an industry standard, open platform for writing applications that work with SAP on the backend. Most Fiori applications have a corresponding SAP Gateway service on the backend. With the latest S/4HANA software, SAP Gateway becomes even more critical for custom application development.

While Fiori is a very powerful platform, some organizations will want to leverage their existing software developer skillsets. For example, on a recent project at a large retailer, they had a team of C# developers, whom they wanted to utilize to build automated testing tools for a new SAP implementation of S/4HANA. Because all of their Fiori applications had an open RESTful SAP Gateway service on the backend, they could call these services directly with any programming language. This tutorial will walk through the steps needed to call SAP Gateway services from any programming language. For this tutorial, we will use Kotlin. Simply substitute any language for the Kotlin code in this tutorial, to utilize the same open RESTful SAP Gateway web services we will build here.

Kotlin is rising very quickly in popularity and Google announced it was their preferred language for Android Development:

https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

Understanding the connectivity lessons learned in this tutorial, should give you a jump start with your application. From there, add your own UI development and any ABAP code on the back end to accomplish any update in SAP (i.e. Sales Order Create BAPI call, etc.). This application is a simple console application to demonstrate all of the CRUD (Create, Read, Update, Delete) operations for a custom table in SAP. While I’m not sharing the code on GitHub, all of the code is in a single file (TableMaintenance.kt), so is easily used in your own project.

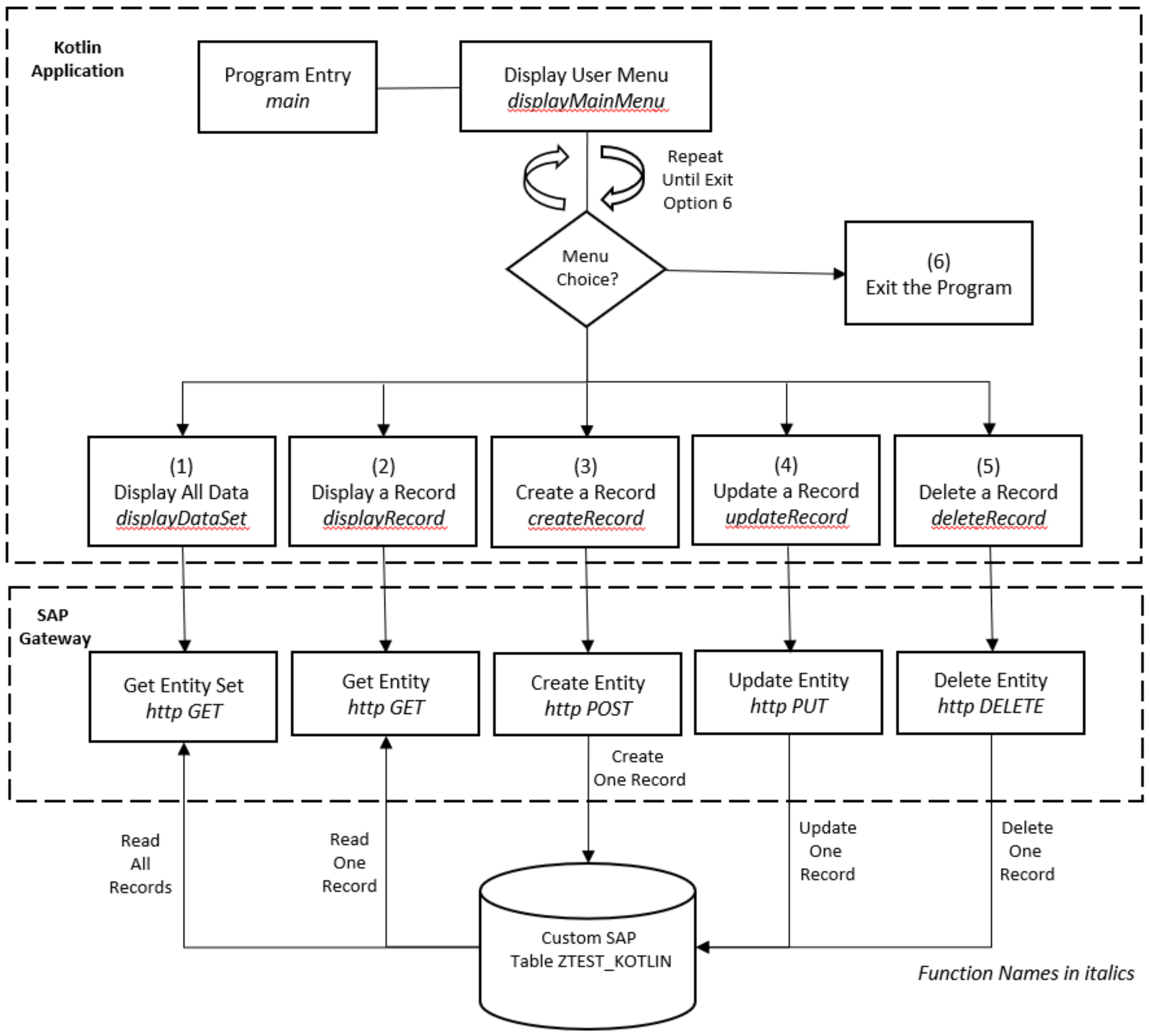
Table of Contents

Software Versions:

- Application Server ABAP 7.54 SP00, S/4HANA, SAPGUI Version 760.
- Eclipse Version: 2019-03 (4.11.0), ABAP Development Tools 3.4.4
- Version 2019.3.2 (Ultimate Edition) of the JetBrains IntelliJ IDEA.

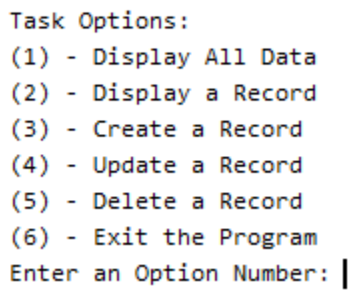
Application Overview

Here is an overview of how the application will work, representing the classical CRUD operations (Create, Read, Update, Delete) on a table:



Kotlin Application
Program Entry *main*
Display User Menu *displayMainMenu*
Repeat Until Exit Option 6
Menu Choice?
(6) Exit the Program
(1) Display All Data *displayDataSet*
(2) Display a Record *displayRecord*
(3) Create a Record *createRecord*
(4) Update a Record *updateRecord*
(5) Delete a Record *deleteRecord*
SAP Gateway
Get Entity Set *http GET*
Get Entity *http GET*
Create Entity *http POST*
Update Entity *http PUT*
Delete Entity *http DELETE*
Create One Record
Read All Records
Read One Record
Update One Record
Delete One Record
Custom SAP Table ZTEST_KOTLIN
Function Names in italics

Options 1-6 will be displayed with the following User Menu in a simple console application:



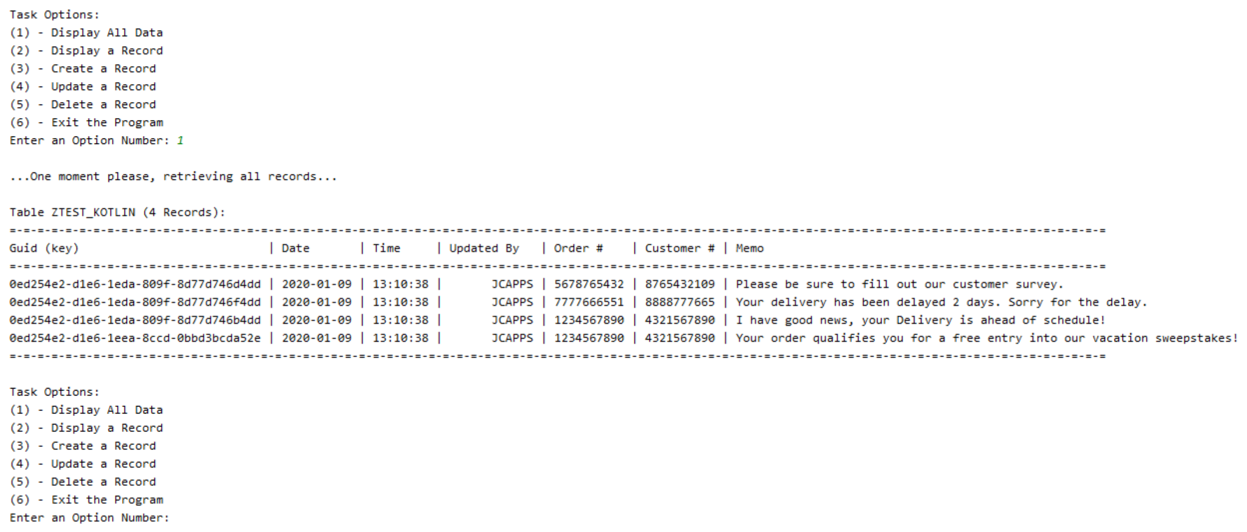
```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number:
```

END USER WALK THROUGH

Before diving into the code, let's do a walk-through of exactly what the simple console application will do.

OPTION 1 – DISPLAY ALL DATA:

Enter option 1, and press enter, will display all data in the custom SAP Table ZTEST_KOTLIN.

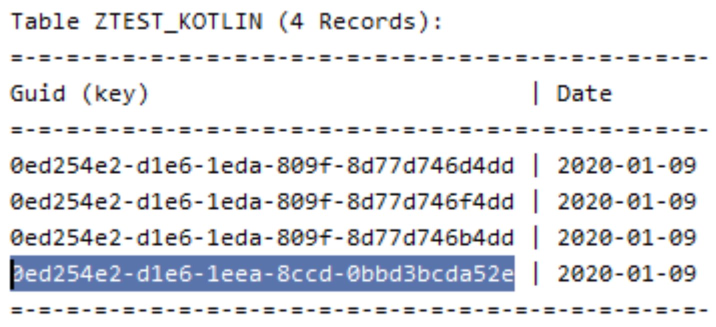


```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 1

...One moment please, retrieving all records...

Table ZTEST_KOTLIN (4 Records):
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Guid (key)                           | Date       | Time     | Updated By | Order #    | Customer # | Memo
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
0ed254e2-d1e6-1eda-809f-8d77d746d4dd | 2020-01-09 | 13:10:38 |     JCAPPS | 5678765432 | 8765432109 | Please be sure to fill out our customer survey.
0ed254e2-d1e6-1eda-809f-8d77d746f4dd | 2020-01-09 | 13:10:38 |     JCAPPS | 7777666551 | 8888777665 | Your delivery has been delayed 2 days. Sorry for the delay.
0ed254e2-d1e6-1eda-809f-8d77d746b4dd | 2020-01-09 | 13:10:38 |     JCAPPS | 1234567890 | 4321567890 | I have good news, your Delivery is ahead of schedule!
0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e | 2020-01-09 | 13:10:38 |     JCAPPS | 1234567890 | 4321567890 | Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number:
```

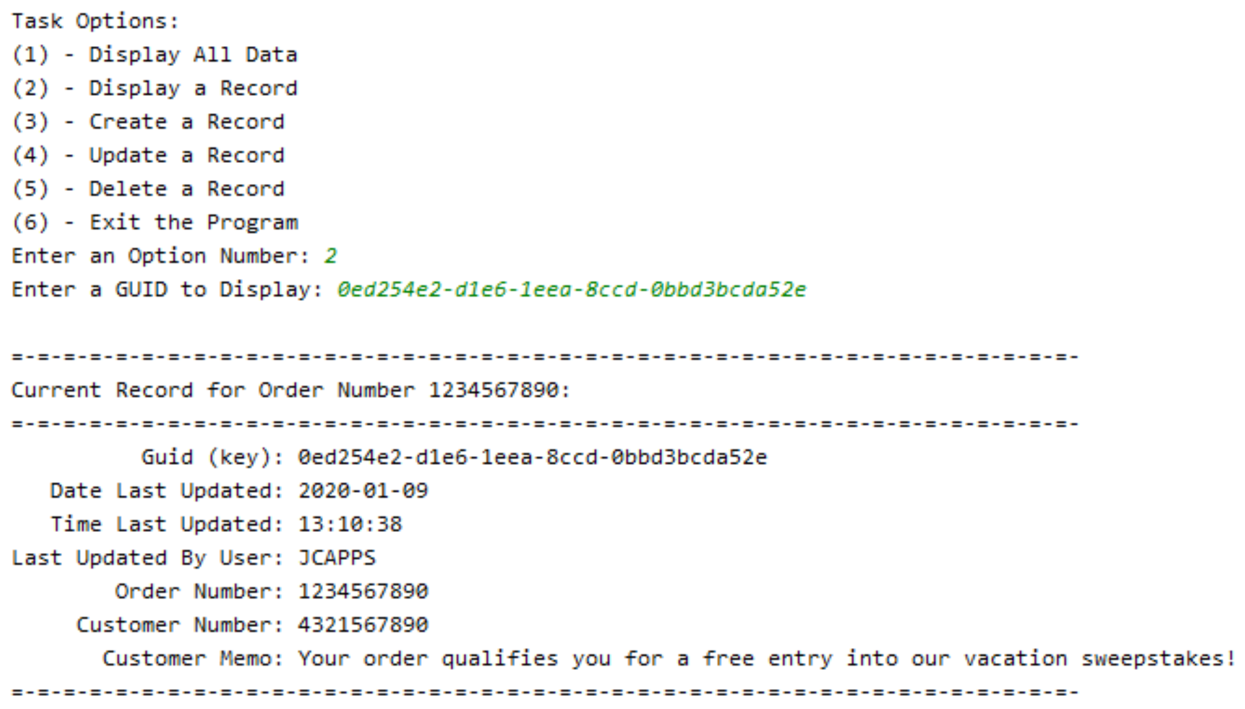
After each option, the main menu is displayed again. Until you enter option 6 (Exit the Program), the main menu will continue to display.

OPTION 2 – DISPLAY A RECORD:

In order to display a single record, highlight and copy one of the Guid keys for a record, as below.



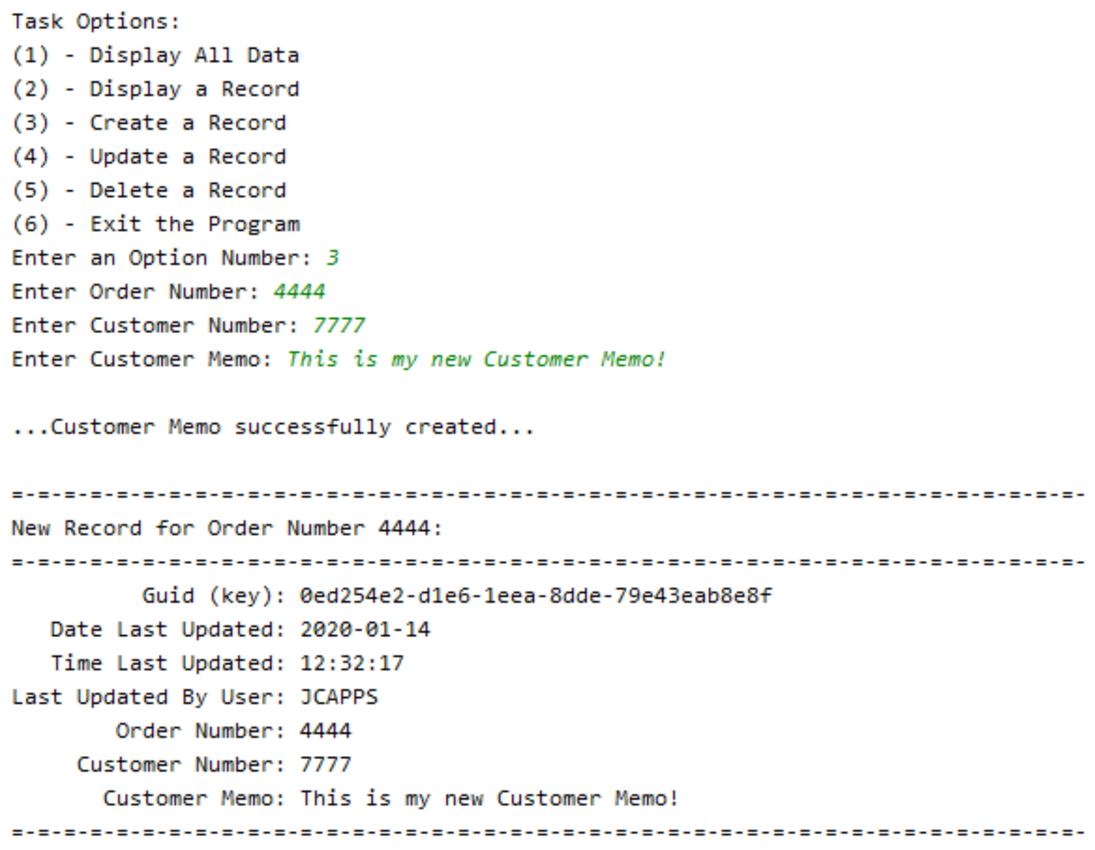
```
Table ZTEST_KOTLIN (4 Records):
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Guid (key)                           | Date
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
0ed254e2-d1e6-1eda-809f-8d77d746d4dd | 2020-01-09
0ed254e2-d1e6-1eda-809f-8d77d746f4dd | 2020-01-09
0ed254e2-d1e6-1eda-809f-8d77d746b4dd | 2020-01-09
0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e | 2020-01-09
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 2
Enter a GUID to Display: 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Current Record for Order Number 1234567890:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e
   Date Last Updated: 2020-01-09
   Time Last Updated: 13:10:38
Last Updated By User: JCAPPS
        Order Number: 1234567890
     Customer Number: 4321567890
       Customer Memo: Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

OPTION 3 – CREATE A RECORD:

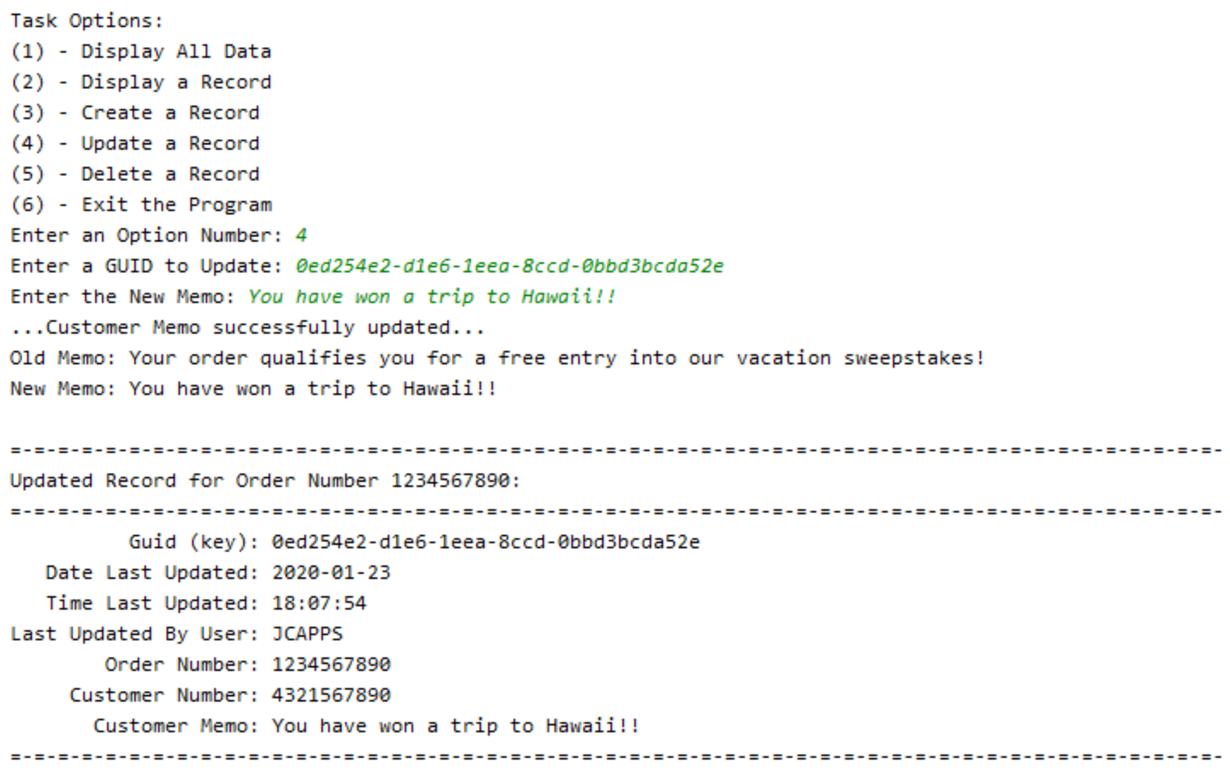
Our intent is to maintain records for Customer Memos for a particular Order and Customer. To create a record, we'll need to enter an Order Number, Customer Number and the Memo. A Guid key is auto generated for us.



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 3
Enter Order Number: 4444
Enter Customer Number: 7777
Enter Customer Memo: This is my new Customer Memo!

...Customer Memo successfully created...

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
New Record for Order Number 4444:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-8dde-79e43eab8e8f
   Date Last Updated: 2020-01-14
   Time Last Updated: 12:32:17
Last Updated By User: JCAPPS
        Order Number: 4444
     Customer Number: 7777
       Customer Memo: This is my new Customer Memo!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

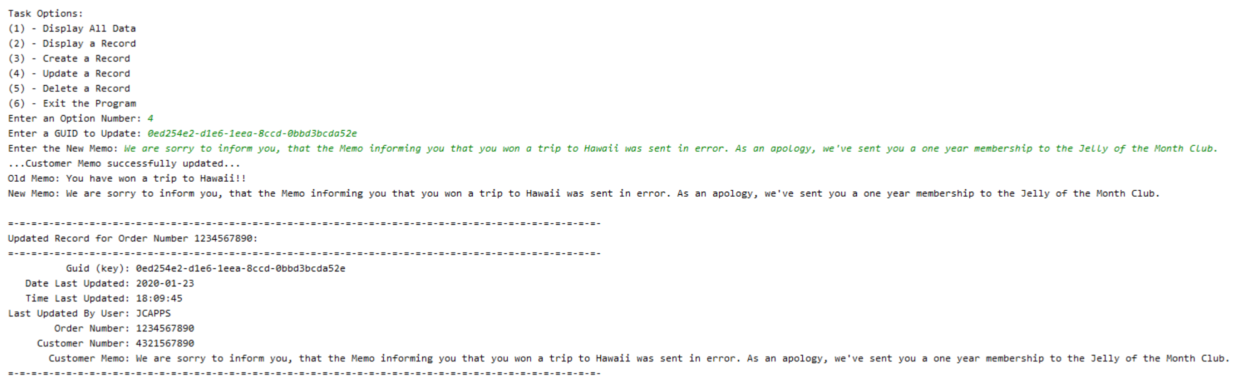
OPTION 4 – UPDATE A RECORD:

To update a record, highlight and copy the Guid key first, then enter the new Customer Memo.



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 4
Enter a GUID to Update: 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e
Enter the New Memo: You have won a trip to Hawaii!!
...Customer Memo successfully updated...
Old Memo: Your order qualifies you for a free entry into our vacation sweepstakes!
New Memo: You have won a trip to Hawaii!!

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Updated Record for Order Number 1234567890:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e
   Date Last Updated: 2020-01-23
   Time Last Updated: 18:07:54
Last Updated By User: JCAPPS
        Order Number: 1234567890
     Customer Number: 4321567890
       Customer Memo: You have won a trip to Hawaii!!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

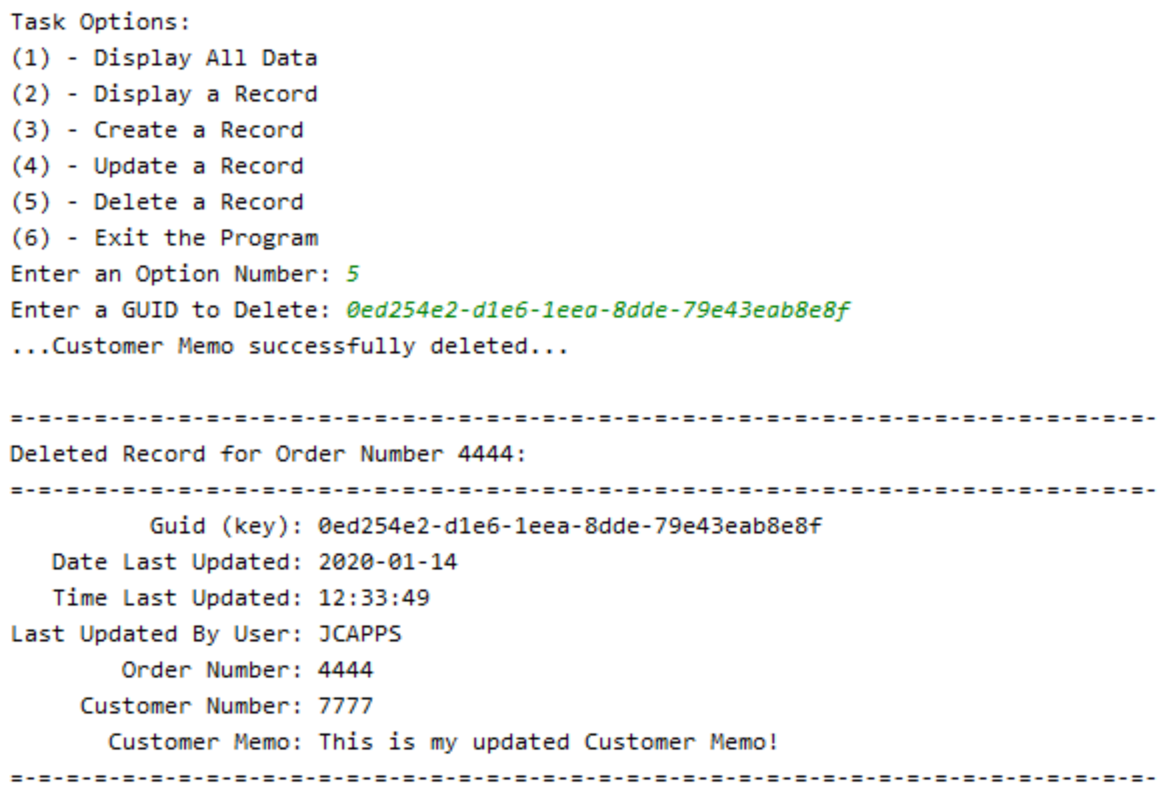
Until you hit option 6 to exit, you can continue to work…



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 4
Enter a GUID to Update: 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e
Enter the New Memo: We are sorry to inform you, that the Memo informing you that you won a trip to Hawaii was sent in error. As an apology, we've sent you a one year membership to the Jelly of the Month Club.
...Customer Memo successfully updated...
Old Memo: You have won a trip to Hawaii!!
New Memo: We are sorry to inform you, that the Memo informing you that you won a trip to Hawaii was sent in error. As an apology, we've sent you a one year membership to the Jelly of the Month Club.

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Updated Record for Order Number 1234567890:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e
   Date Last Updated: 2020-01-23
   Time Last Updated: 18:09:45
Last Updated By User: JCAPPS
        Order Number: 1234567890
     Customer Number: 4321567890
       Customer Memo: We are sorry to inform you, that the Memo informing you that you won a trip to Hawaii was sent in error. As an apology, we've sent you a one year membership to the Jelly of the Month Club.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

OPTION 5 – DELETE A RECORD:

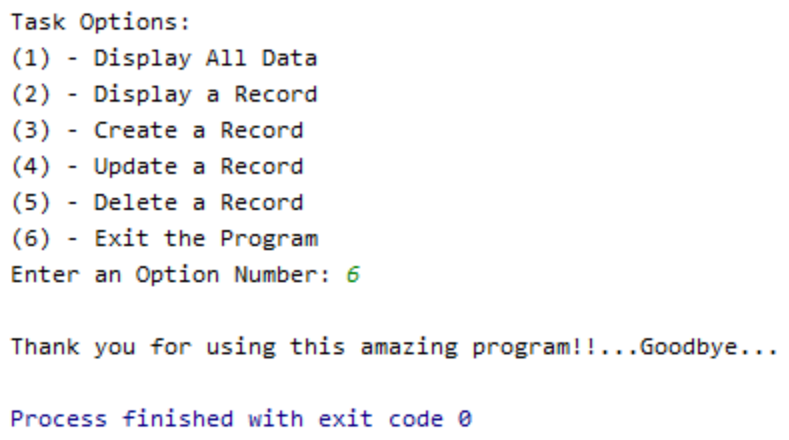
Just like with an Update, copy the Guid key first, in order to delete a record. Let's delete the above record we just created and updated.



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 5
Enter a GUID to Delete: 0ed254e2-d1e6-1eea-8dde-79e43eab8e8f
...Customer Memo successfully deleted...

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Deleted Record for Order Number 4444:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-8dde-79e43eab8e8f
   Date Last Updated: 2020-01-14
   Time Last Updated: 12:33:49
Last Updated By User: JCAPPS
        Order Number: 4444
     Customer Number: 7777
       Customer Memo: This is my updated Customer Memo!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

OPTION 6 – EXIT THE PROGRAM:

Option 6 simply ends the program.



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 6

Thank you for using this amazing program!!...Goodbye...

Process finished with exit code 0
```

Now that we see how the program works, let's write the code to accomplish all of the above…

Create an SAP Gateway Service

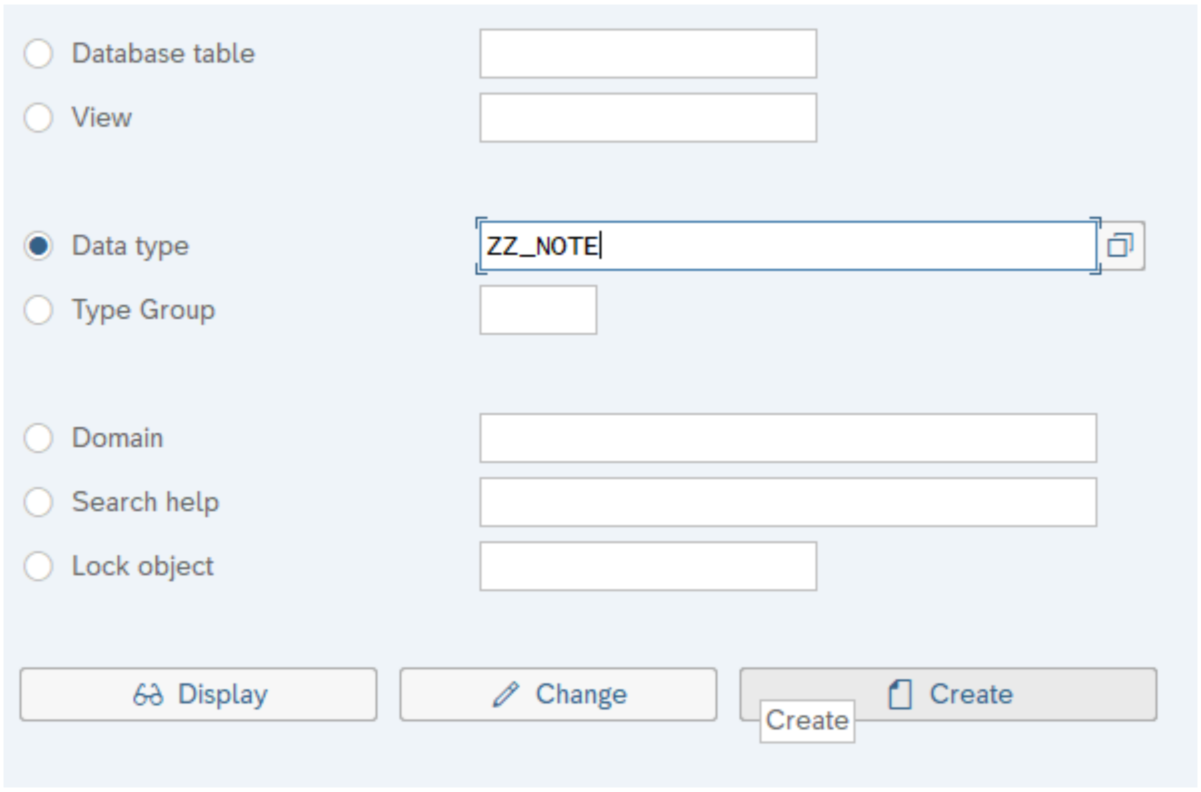
CREATE THE CUSTOM TABLE

First, we will create our custom table in SAP named ZTEST_KOTLIN. You can either perform the below steps, or follow this SAP Excel Blog, which will also allow you to create some test records using Excel:

https://blogs.sap.com/2019/11/13/excel-xlsx-table-maintenance/

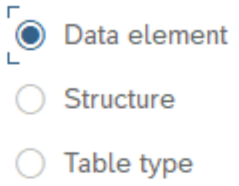
First, let's create a new String Data Element, which we will use for our Customer Memo field.

Enter the SAP Transaction SE11, and enter the Data type "ZZ_NOTE", and click the "Create" button:



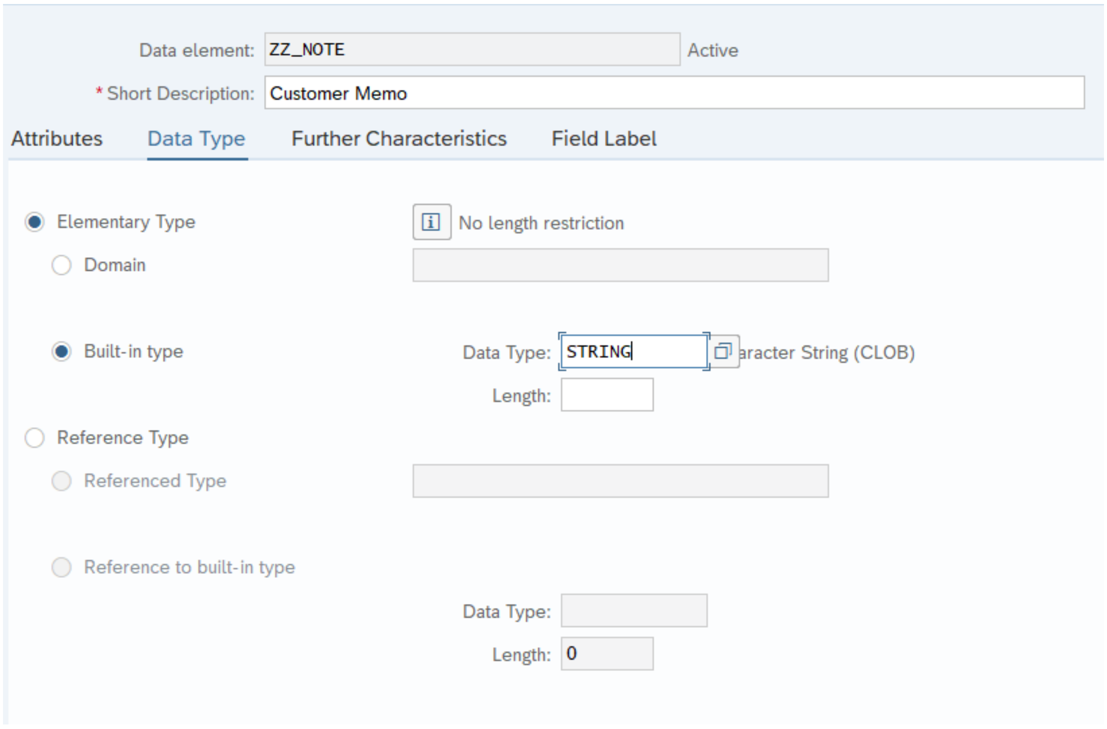
Database table [ ]
View [ ]
[✓ Data type] ZZ_NOTE
Type Group [ ]
Domain [ ]
Search help [ ]
Lock object [ ]

Select "Data Element" and continue:



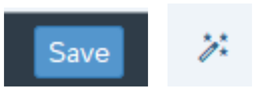
[✓ Data element]
Structure
Table type

Enter the following, then Save and Activate:

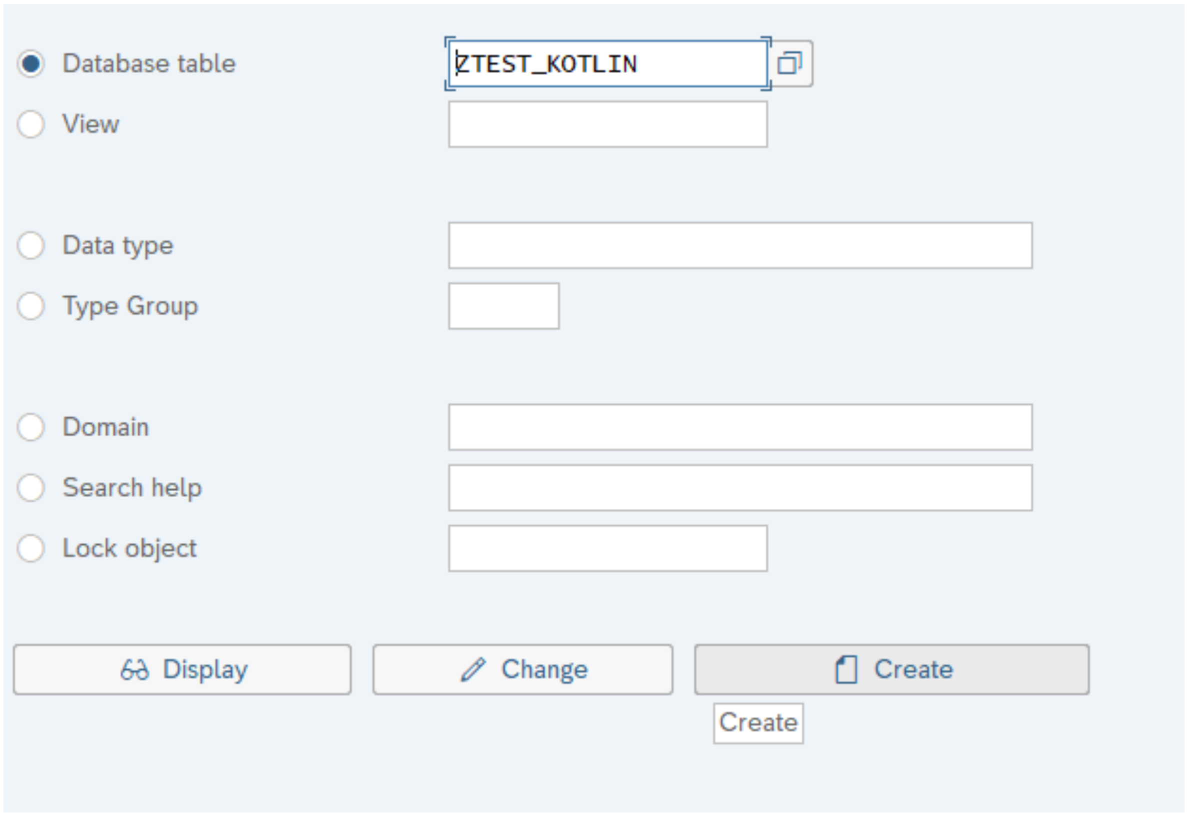

Data element: ZZ_NOTE Active

* Short Description: Customer Memo

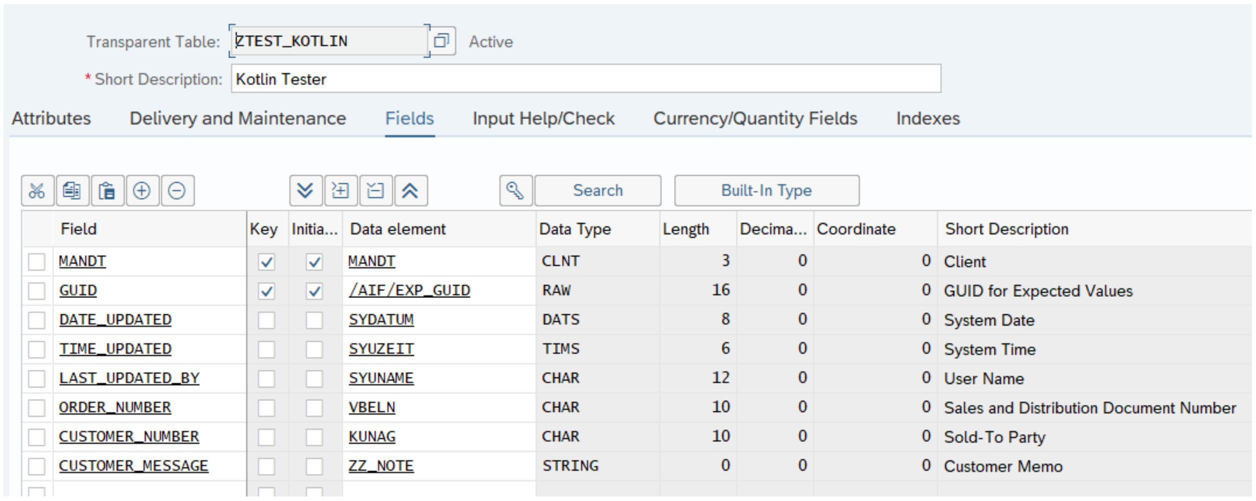
(●) Elementary Type
No length restriction
( ) Domain
(●) Built-in type
Data Type: STRING aracter String (CLOB)
Length:
( ) Reference Type
( ) Referenced Type
( ) Reference to built-in type
Data Type:
Length: 0



Execute the SAP Transaction SE11, and enter table name “ZTEST_KOTLIN”, then click on the “Create” button:


(●) Database table: ZTEST_KOTLIN
( ) View
( ) Data type
( ) Type Group
( ) Domain
( ) Search help
( ) Lock object

Enter the following fields for ZTEST_KOTLIN, and Save and Activate:



Transparent Table: ZTEST_KOTLIN Active

* Short Description: Kotlin Tester

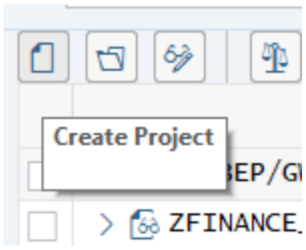
Field	Key	Initia...	Data element	Data Type	Length	Decima...	Coordinate	Short Description
MANDT	✓	✓	MANDT	CLNT	3	0	0	Client
GUID	✓	✓	/AIF/EXP_GUID	RAW	16	0	0	GUID for Expected Values
DATE_UPDATED			SYDATUM	DATS	8	0	0	System Date
TIME_UPDATED			SYUZEIT	TIMS	6	0	0	System Time
LAST_UPDATED_BY			SYUNAME	CHAR	12	0	0	User Name
ORDER_NUMBER			VBELN	CHAR	10	0	0	Sales and Distribution Document Number
CUSTOMER_NUMBER			KUNAG	CHAR	10	0	0	Sold-To Party
CUSTOMER_MESSAGE			ZZ_NOTE	STRING	0	0	0	Customer Memo

If you prefer Eclipse, we can create the table by entering the following DDL for our Table:

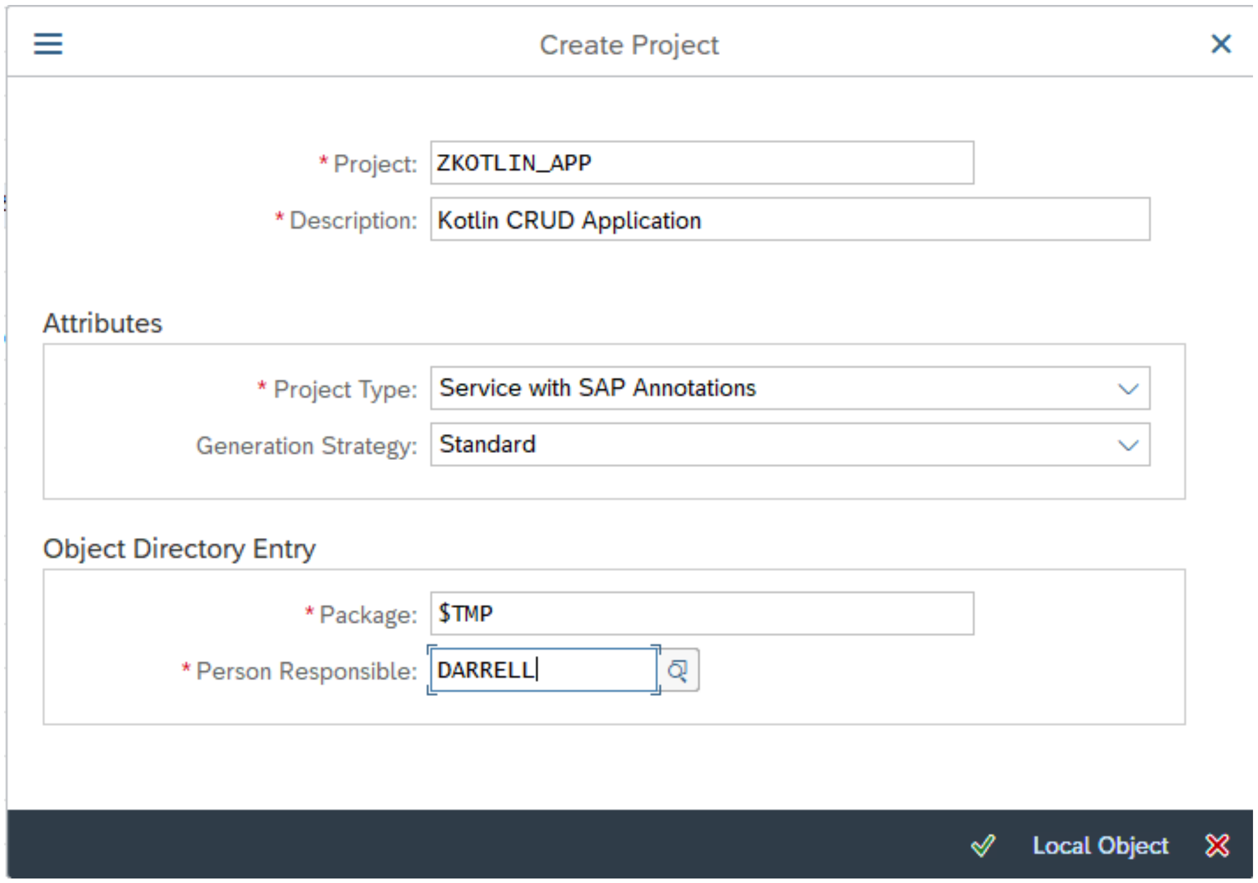
```
@EndUserText.label : 'Kotlin Tester'
@AbapCatalog.enhancementCategory : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #ALLOWED
define table ztest_kotlin {
  key mandt         : mandt not null;
  key guid          : /aif/exp_guid not null;
  date_updated      : sydatum;
  time_updated      : syuzeit;
  last_updated_by   : syuname;
  order_number      : vbeln;
  customer_number   : kunag;
  customer_message  : zz_note;

}
```

CREATE A NEW GATEWAY PROJECT

Execute the SAP Transaction SEGW and click on the “Create Project” button:



Enter the following and continue:


Create Project

* Project:	ZKOTLIN_APP
* Description:	Kotlin CRUD Application

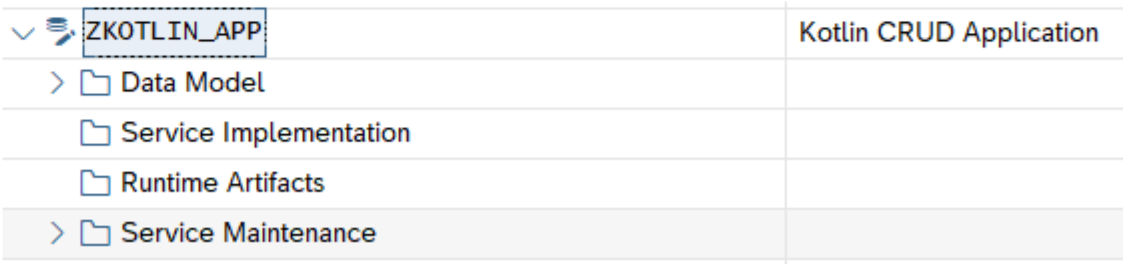
Attributes

* Project Type:	Service with SAP Annotations
Generation Strategy:	Standard

Object Directory Entry

* Package:	$TMP
* Person Responsible:	DARRELL

We now have the following tree structure:


ZKOTLIN_APP	Kotlin CRUD Application
Data Model	
Service Implementation	
Runtime Artifacts	
Service Maintenance	

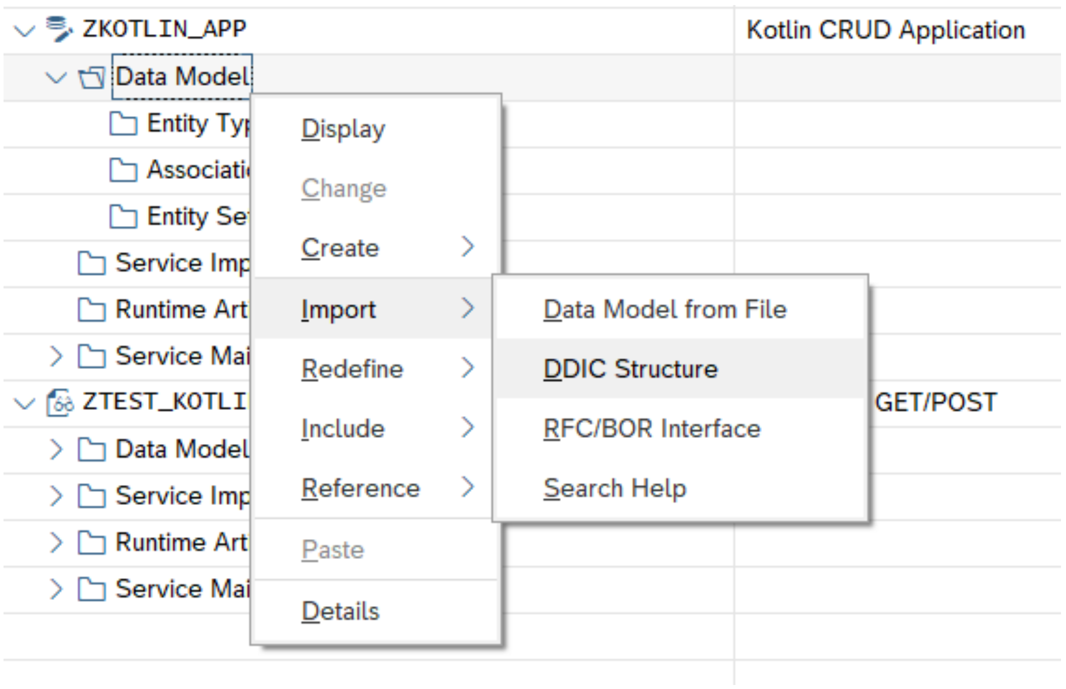
At this point, there are many ways to setup a basic CRUD Gateway Service. Among them is BOPF, which auto-generates code that allows you to easily maintain a Z Table via CDS View Annotations. Another method is to setup a CDS View, which we expose as a service to quickly enable a “GET” or Read operation.

As I recently witnessed at SAP Teched on newer versions of SAP, you can even right-click on your service, and auto-generate a Fiori App which displays your table. If your application evolves into something complicated, the road will lead to a more “custom” approach I will show in this Blog.

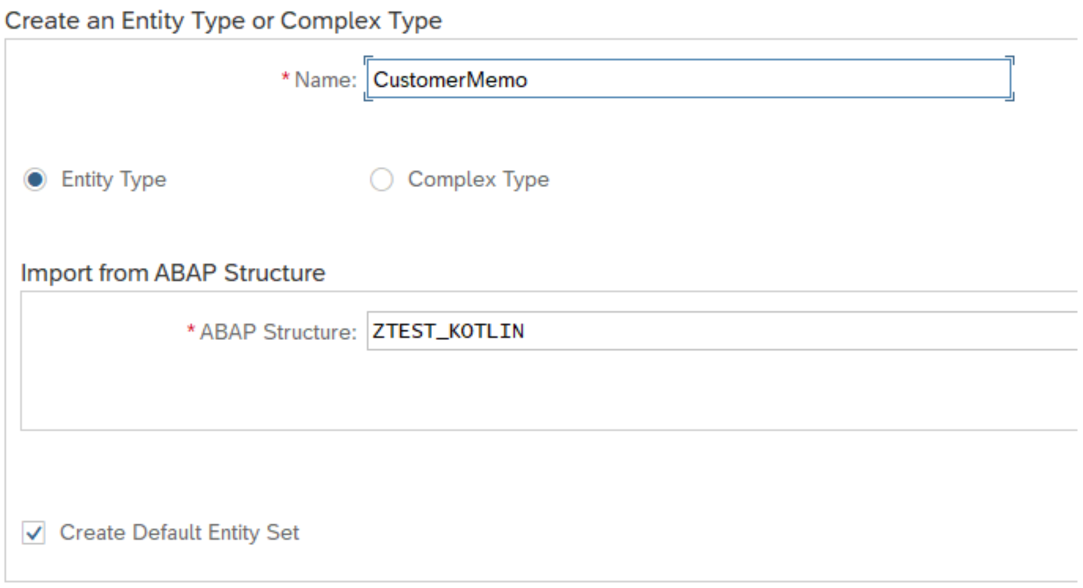
It’s usually not long, before you need to start doing a “Deep Entity” POST, for example a Sales Order create with header and line item information, where the code generation method quickly reaches it’s limitation. Plus, if you are using Kotlin, let's say to write a native Android Application, you may not be using Fiori at all. SAP is marketing a "Bring Your Own Language" (BYOL) approach with HANA.

Yes, there are many plug-and-play options for a CRUD application, but the entire point of BYOL, is so organizations can utilize their existing pool of software engineers. For example, if you have a team of Kotlin developers, you may want to utilize them with your SAP installation. So, let's proceed with a bare bones custom approach that gives us maximum flexibility, without the use of any code generation restrictions...

Let's define our Entity, which represents the attributes of the ZTEST_KOTLIN table. Right-click on the "Data Model" node, and choose "Import > DDIC Structure":



Let's give our Entity an intuitive name, like "CustomerMemo". Enter the following, and click "Next":



Create an Entity Type or Complex Type

* Name: CustomerMemo

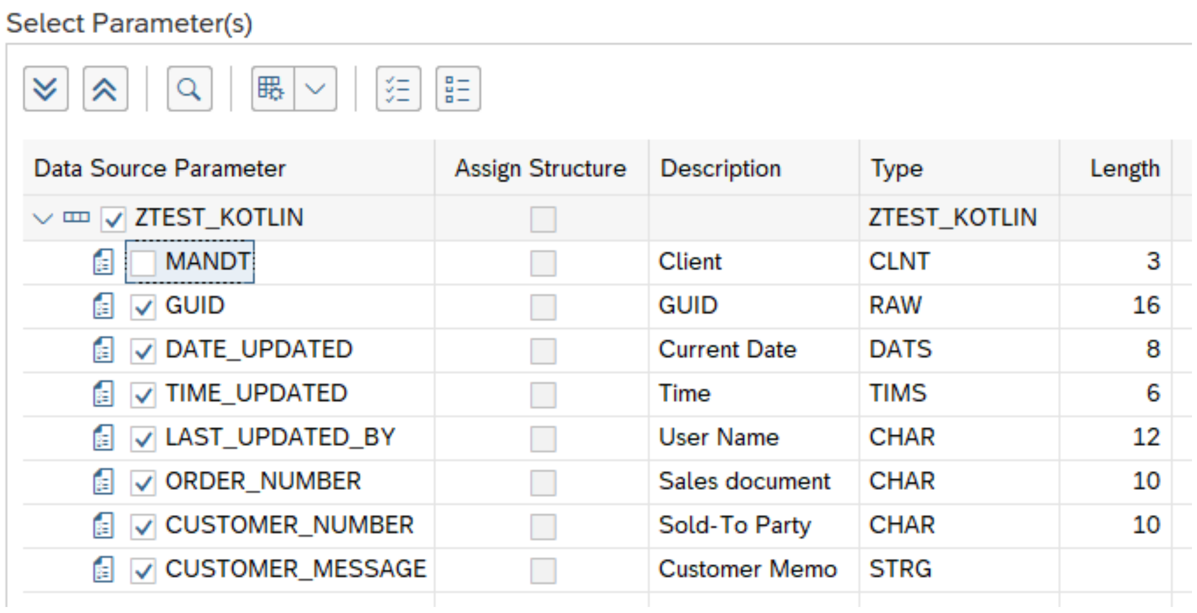
[✓ Entity Type] Complex Type

Import from ABAP Structure

* ABAP Structure: ZTEST_KOTLIN

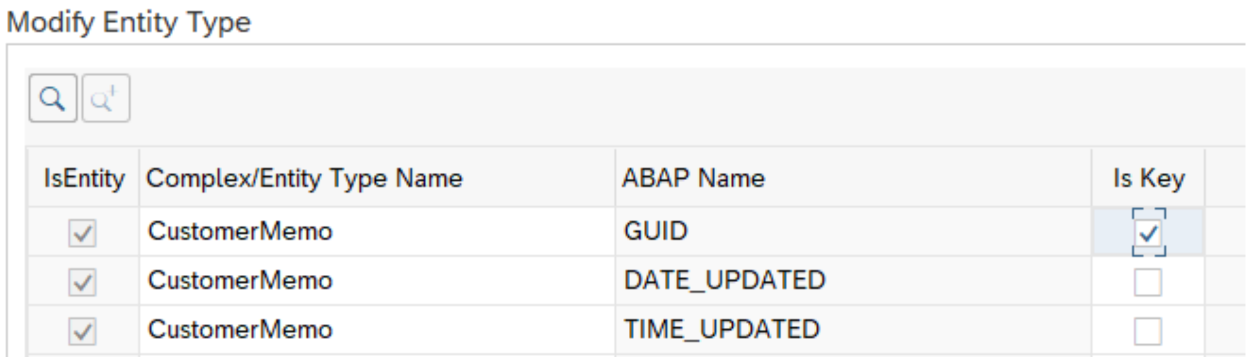
☑ Create Default Entity Set

Select all of the fields, except "MANDT", and click "Next":


Select Parameter(s)

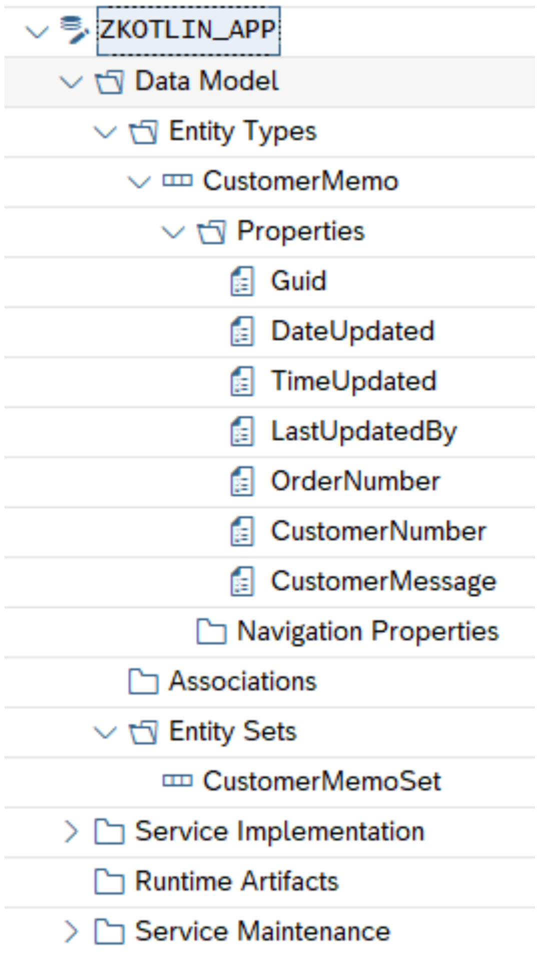
Data Source Parameter	Assign Structure	Description	Type	Length
✓ ZTEST_KOTLIN	☐		ZTEST_KOTLIN	
☐ MANDT	☐	Client	CLNT	3
✓ GUID	☐	GUID	RAW	16
✓ DATE_UPDATED	☐	Current Date	DATS	8
✓ TIME_UPDATED	☐	Time	TIMS	6
✓ LAST_UPDATED_BY	☐	User Name	CHAR	12
✓ ORDER_NUMBER	☐	Sales document	CHAR	10
✓ CUSTOMER_NUMBER	☐	Sold-To Party	CHAR	10
✓ CUSTOMER_MESSAGE	☐	Customer Memo	STRG	

Select the GUID field as the key, then click "Finish":

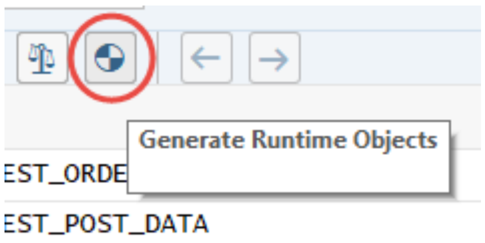

Modify Entity Type

IsEntity	Complex/Entity Type Name	ABAP Name	Is Key
✓	CustomerMemo	GUID	✓
✓	CustomerMemo	DATE_UPDATED	☐
✓	CustomerMemo	TIME_UPDATED	☐

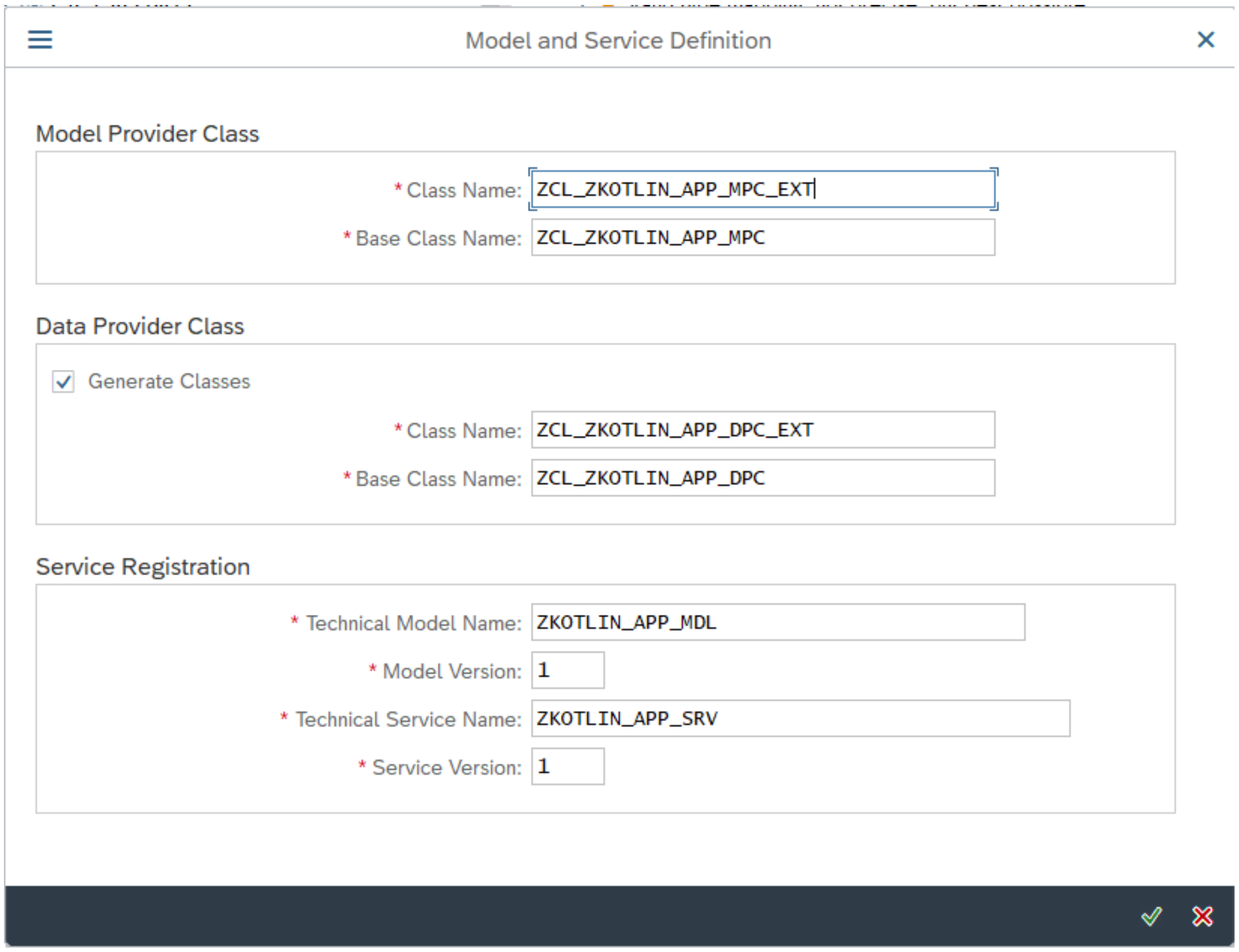
Now if you expand out the Entity node, you can see your newly defined Properties, and the auto-generated Entity Set:



Save your new Data Model, and click on the "Generate Runtime Objects" button:



Accept the proposed defaults for our class names, and click Continue (Enter):


Model and Service Definition

Model Provider Class

* Class Name:	ZCL_ZKOTLIN_APP_MPC_EXT
* Base Class Name:	ZCL_ZKOTLIN_APP_MPC

Data Provider Class

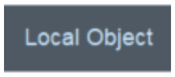
☑ Generate Classes

* Class Name:	ZCL_ZKOTLIN_APP_DPC_EXT
* Base Class Name:	ZCL_ZKOTLIN_APP_DPC

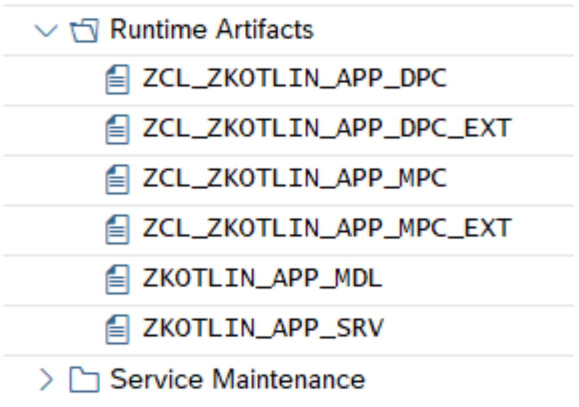
Service Registration

* Technical Model Name:	ZKOTLIN_APP_MDL
* Model Version:	1
* Technical Service Name:	ZKOTLIN_APP_SRV
* Service Version:	1

Click on the “Local Object” button on the next pop-up to complete (assuming you don't want to transport it).



Now, navigate to the below node, to see the new Runtime Artifacts which were generated by the SAP Gateway Service Builder:


Runtime Artifacts
- ZCL_ZKOTLIN_APP_DPC
- ZCL_ZKOTLIN_APP_DPC_EXT
- ZCL_ZKOTLIN_APP_MPC
- ZCL_ZKOTLIN_APP_MPC_EXT
- ZKOTLIN_APP_MDL
- ZKOTLIN_APP_SRV

Service Maintenance

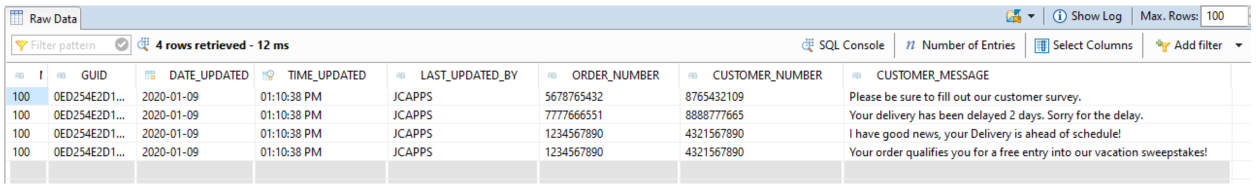
The *MPC* classes handle the data model definition and setup. The *DPC* classes handle the population of the entities with data. To keep it simple, you will always write code in the *DPC_EXT class. In the following sections, we will write code in the methods of our ZCL_ZKOTLIN_APP_DPC_EXT class to perform all of the CRUD operations.

READ RECORDS

In order to setup our table Read operations, it would be nice to have some test data in the ZTEST_KOTLIN table. Because the Guid key field is auto-generated, that could be a little complicated. The easiest way, would be to use the ABAP code from this Blog, and enter some test data using Excel:

https://blogs.sap.com/2019/11/13/excel-xlsx-table-maintenance/

For this example, I've created the following 4 records displayed in Eclipse, below:

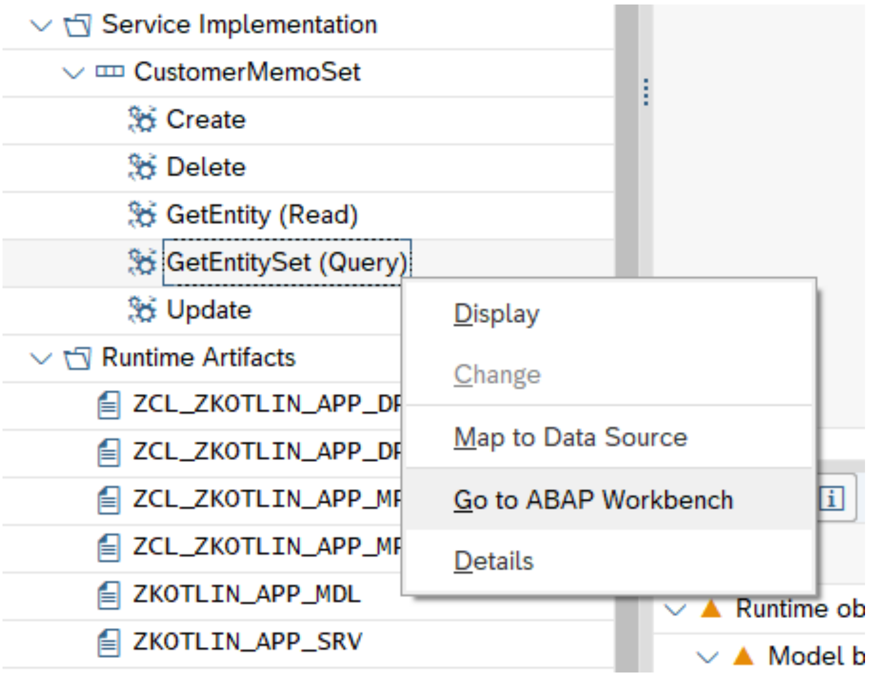


I	GUID	DATE_UPDATED	TIME_UPDATED	LAST_UPDATED_BY	ORDER_NUMBER	CUSTOMER_NUMBER	CUSTOMER_MESSAGE
100	0ED254E2D1...	2020-01-09	01:10:38 PM	JCAPPS	5678765432	8765432109	Please be sure to fill out our customer survey.
100	0ED254E2D1...	2020-01-09	01:10:38 PM	JCAPPS	7777666551	8888777665	Your delivery has been delayed 2 days. Sorry for the delay.
100	0ED254E2D1...	2020-01-09	01:10:38 PM	JCAPPS	1234567890	4321567890	I have good news, your Delivery is ahead of schedule!
100	0ED254E2D1...	2020-01-09	01:10:38 PM	JCAPPS	1234567890	4321567890	Your order qualifies you for a free entry into our vacation sweepstakes!

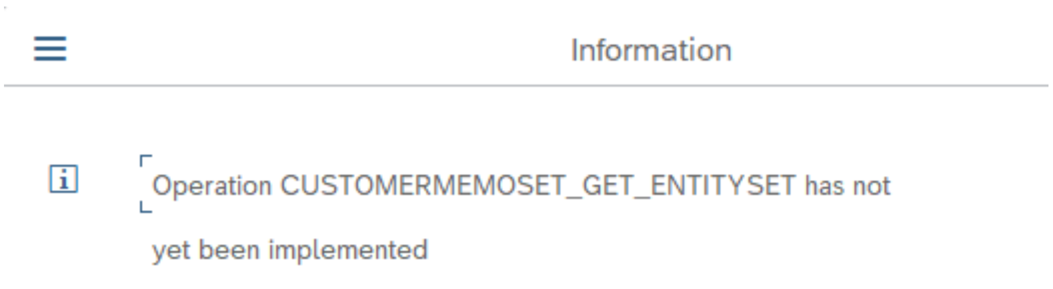
Alternatively, you can display the table records in the SAP transaction SE16. You will see the above 4 records used in the remainder of this Blog.

Read All Records

In order to read the complete table, and pass it back, we will implement the "GetEntitySet" Service. Navigate to the node "Service Implementation > GetEntitySet" and right-click and select "Go to ABAP Workbench":



You'll get a pop-up, which tells you that you still have to write some code in the method:

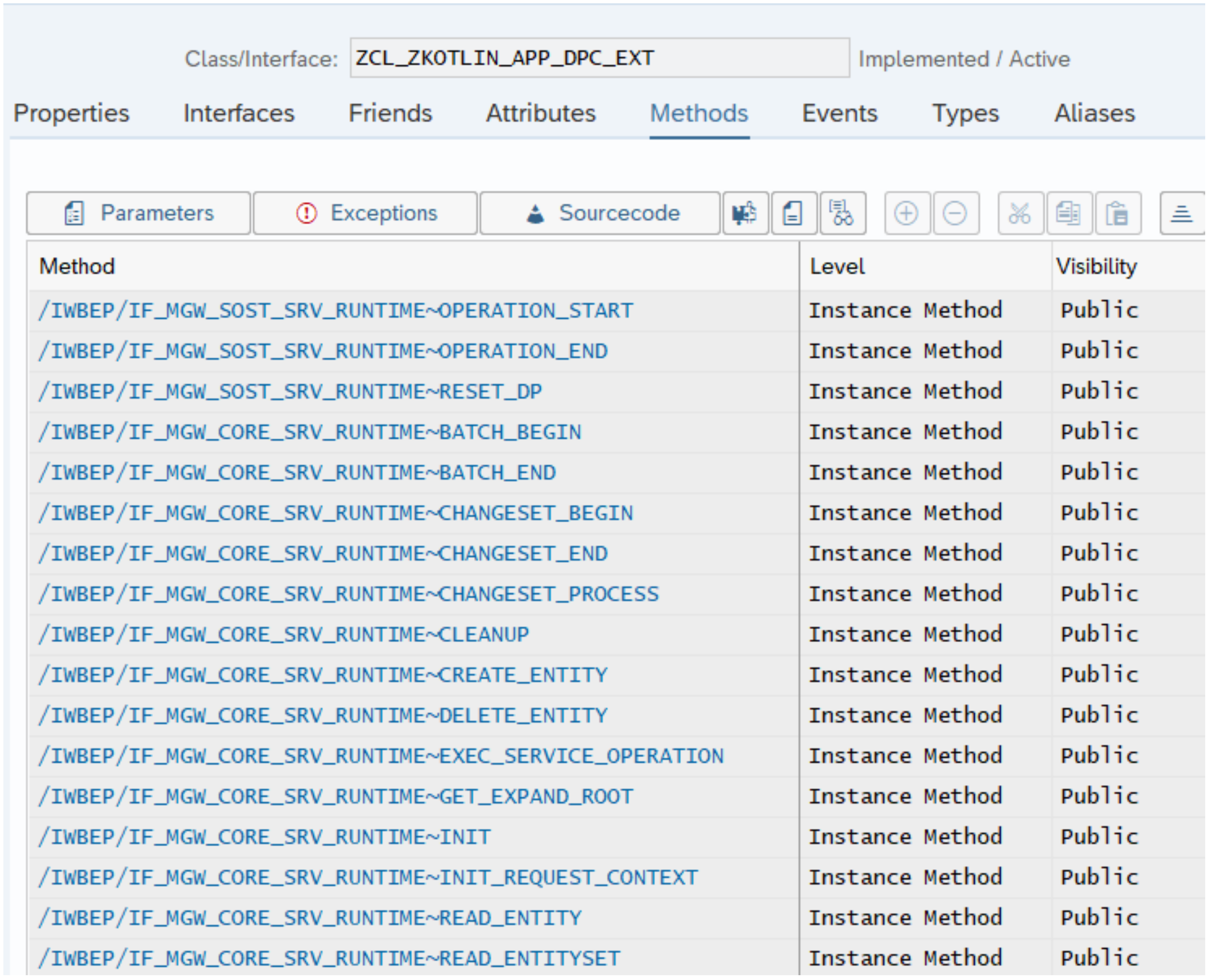


Information

Operation CUSTOMERMEMOSET_GET_ENTITYSET has not yet been implemented

Hit Enter to continue.

This will take us to the ABAP Class Editor, where you can see lots of methods that were created by the SAP Gateway Service Builder:

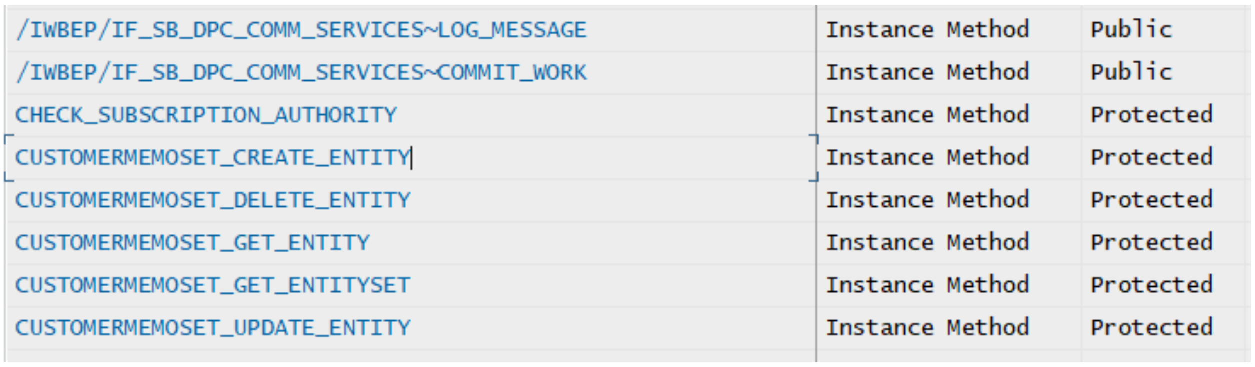


Class/Interface: ZCL_ZKOTLIN_APP_DPC_EXT Implemented / Active

Method	Level	Visibility
/IWBEP/IF_MGW_SOST_SRV_RUNTIME~OPERATION_START	Instance Method	Public
/IWBEP/IF_MGW_SOST_SRV_RUNTIME~OPERATION_END	Instance Method	Public
/IWBEP/IF_MGW_SOST_SRV_RUNTIME~RESET_DP	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~BATCH_BEGIN	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~BATCH_END	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~CHANGESET_BEGIN	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~CHANGESET_END	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~CHANGESET_PROCESS	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~CLEANUP	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~CREATE_ENTITY	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~DELETE_ENTITY	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~EXEC_SERVICE_OPERATION	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~GET_EXPAND_ROOT	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~INIT	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~INIT_REQUEST_CONTEXT	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~READ_ENTITY	Instance Method	Public
/IWBEP/IF_MGW_CORE_SRV_RUNTIME~READ_ENTITYSET	Instance Method	Public

In order to write our custom code, we need to do a Redefinition on the SAP generated methods. This is would be similar to a method Override in Java.

Page down to the bottom of the method list, and you will see the methods we will implement to support all of our CRUD operations:



/IWBEP/IF_SB_DPC_COMM_SERVICES~LOG_MESSAGE	Instance Method	Public
/IWBEP/IF_SB_DPC_COMM_SERVICES~COMMIT_WORK	Instance Method	Public
CHECK_SUBSCRIPTION_AUTHORITY	Instance Method	Protected
CUSTOMERMEMOSET_CREATE_ENTITY	Instance Method	Protected
CUSTOMERMEMOSET_DELETE_ENTITY	Instance Method	Protected
CUSTOMERMEMOSET_GET_ENTITY	Instance Method	Protected
CUSTOMERMEMOSET_GET_ENTITYSET	Instance Method	Protected
CUSTOMERMEMOSET_UPDATE_ENTITY	Instance Method	Protected

To summarize, we will Redefine the following methods:

CUSTOMERMEMOSET_CREATE_ENTITY: Create a record, HTTP POST

CUSTOMERMEMOSET_DELETE_ENTITY: Delete a record, HTTP DELETE

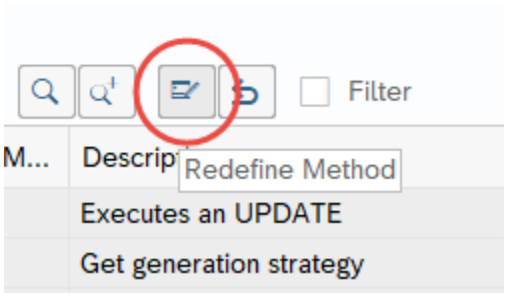
CUSTOMERMEMOSET_GET_ENTITY: Read a single record, HTTP GET

CUSTOMERMEMOSET_GET_ENTITYSET: Read the entire table, and return all records, HTTP GET

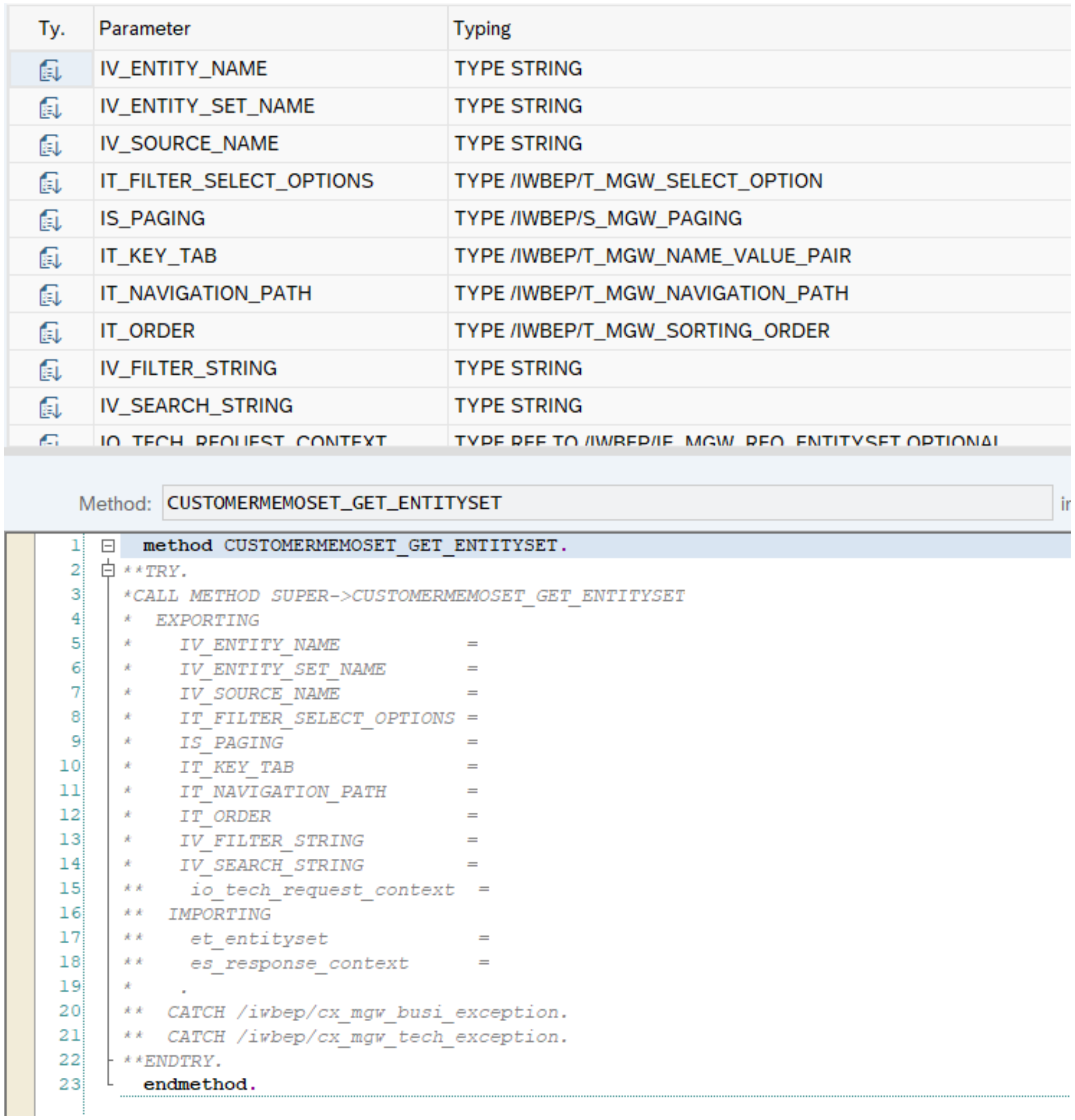
CUSTOMERMEMOSET_UPDATE_ENTITY: Update a record, HTTP PUT

Click on the Change button, to change the class:

Click on the CUSTOMERMEMOSET_GET_ENTITYSET method to place your cursor there, then click on the "Redefine Method" button on the upper-right:



This will open up the method as editable, where you will see some commented out template code:



Ty.	Parameter	Typing
	IV_ENTITY_NAME	TYPE STRING
	IV_ENTITY_SET_NAME	TYPE STRING
	IV_SOURCE_NAME	TYPE STRING
	IT_FILTER_SELECT_OPTIONS	TYPE /IWBEP/T_MGW_SELECT_OPTION
	IS_PAGING	TYPE /IWBEP/S_MGW_PAGING
	IT_KEY_TAB	TYPE /IWBEP/T_MGW_NAME_VALUE_PAIR
	IT_NAVIGATION_PATH	TYPE /IWBEP/T_MGW_NAVIGATION_PATH
	IT_ORDER	TYPE /IWBEP/T_MGW_SORTING_ORDER
	IV_FILTER_STRING	TYPE STRING
	IV_SEARCH_STRING	TYPE STRING
	IO_TECH_REQUEST_CONTEXT	TYPE REF TO /IWBEP/IF_MGW_REQ_ENTITYSET_OPTIONAL

Method: CUSTOMERMEMOSET_GET_ENTITYSET

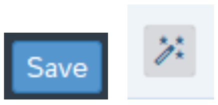
```
method CUSTOMERMEMOSET_GET_ENTITYSET.
**TRY.
*CALL METHOD SUPER->CUSTOMERMEMOSET_GET_ENTITYSET
*  EXPORTING
*    IV_ENTITY_NAME           =
*    IV_ENTITY_SET_NAME       =
*    IV_SOURCE_NAME           =
*    IT_FILTER_SELECT_OPTIONS =
*    IS_PAGING                =
*    IT_KEY_TAB               =
*    IT_NAVIGATION_PATH       =
*    IT_ORDER                 =
*    IV_FILTER_STRING         =
*    IV_SEARCH_STRING         =
**    io_tech_request_context  =
**  IMPORTING
**    et_entityset             =
**    es_response_context      =
*    .
**  CATCH /iwbep/cx_mgw_busi_exception.
**  CATCH /iwbep/cx_mgw_tech_exception.
**ENDTRY.
  endmethod.
```

Delete this code, and insert the following code:

```
method CUSTOMERMEMOSET_GET_ENTITYSET.
  SELECT * FROM ztest_kotlin INTO TABLE et_entityset.
  SORT et_entityset BY date_updated DESCENDING time_updated DESCENDING.
endmethod.
```

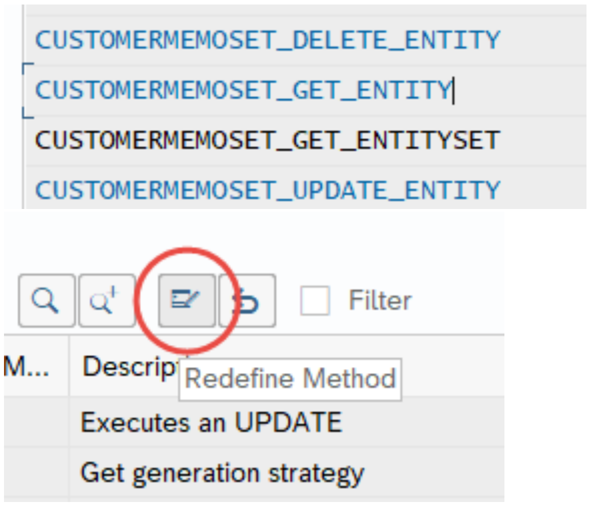
This simply selects all records in the table and returns them in the et_entityset internal table.

Save and Activate this code.



Read a Single Record

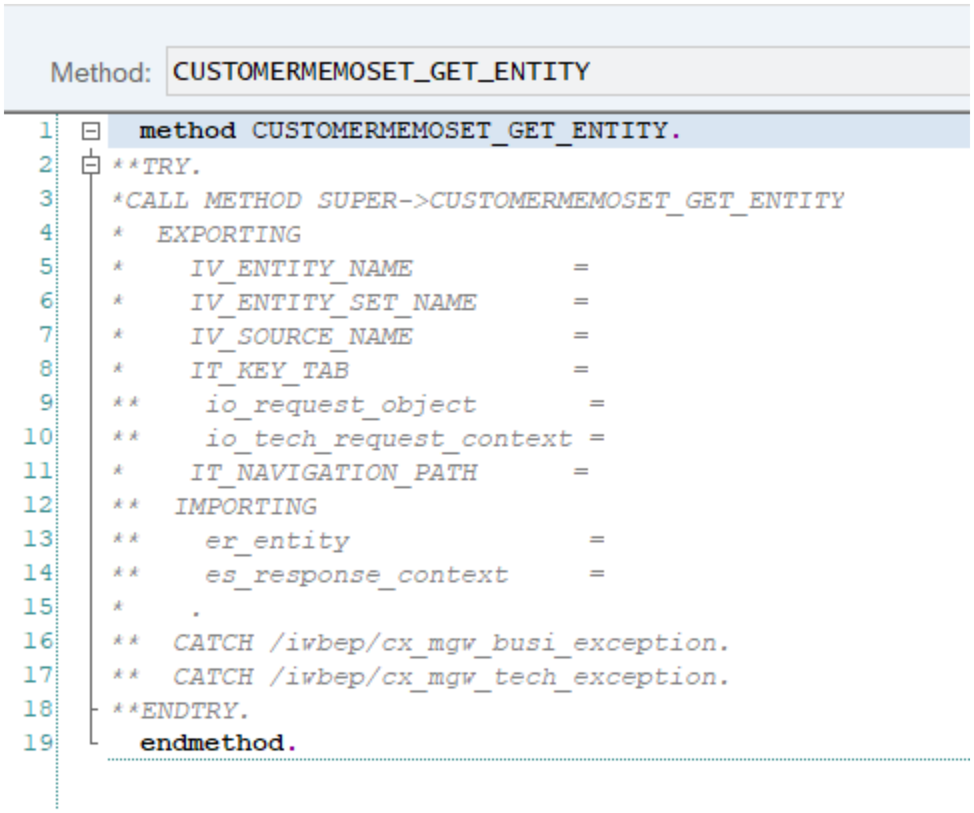
The above method will select all records. We'll also need a method to select individual records. Go back to the method list, and place your cursor in the CUSTOMERMEMOSET_GET_ENTITY method, and click on the "Redefine Method" button, just as we did above:



CUSTOMERMEMOSET_DELETE_ENTITY
CUSTOMERMEMOSET_GET_ENTITY
CUSTOMERMEMOSET_GET_ENTITYSET
CUSTOMERMEMOSET_UPDATE_ENTITY

Executes an UPDATE
Get generation strategy

Delete the commented code in the method:



Method: CUSTOMERMEMOSET_GET_ENTITY

```
  method CUSTOMERMEMOSET_GET_ENTITY.
**TRY.
*CALL METHOD SUPER->CUSTOMERMEMOSET_GET_ENTITY
*  EXPORTING
*    IV_ENTITY_NAME          =
*    IV_ENTITY_SET_NAME      =
*    IV_SOURCE_NAME          =
*    IT_KEY_TAB              =
**    io_request_object       =
**    io_tech_request_context =
*    IT_NAVIGATION_PATH      =
**  IMPORTING
**    er_entity               =
**    es_response_context     =
*    .
**  CATCH /iwbep/cx_mgw_busi_exception.
**  CATCH /iwbep/cx_mgw_tech_exception.
**ENDTRY.
  endmethod.
```

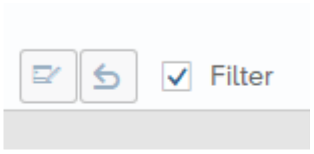
Replace it with the following code:

```
method CUSTOMERMEMOSET_GET_ENTITY.
  DATA: ls_entity LIKE er_entity.

  io_tech_request_context->get_converted_keys( IMPORTING es_key_values = ls_entity ).

  SELECT SINGLE * FROM ztest_kotlin INTO er_entity
     WHERE guid  = ls_entity-guid.
endmethod.
```

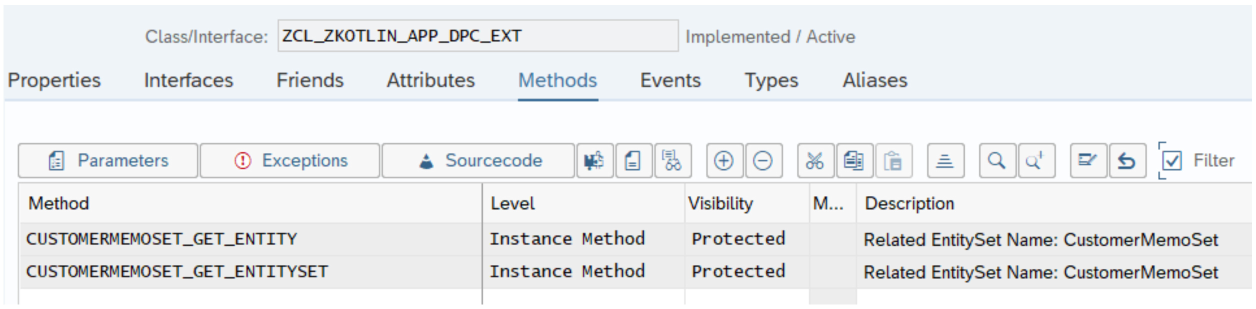
Save and Activate this code.

The above code will pull out the key passed from the application, and fetch a single record from the ZTEST_KOTLIN table.

Now that we have redefined a couple of methods, note the "Filter" checkbox in the upper-right:



Click on this checkbox, to filter out all of the other methods in the class, and show only those we've redefined thus far:



Class/Interface: ZCL_ZKOTLIN_APP_DPC_EXT Implemented / Active

Properties Interfaces Friends Attributes Methods Events Types Aliases

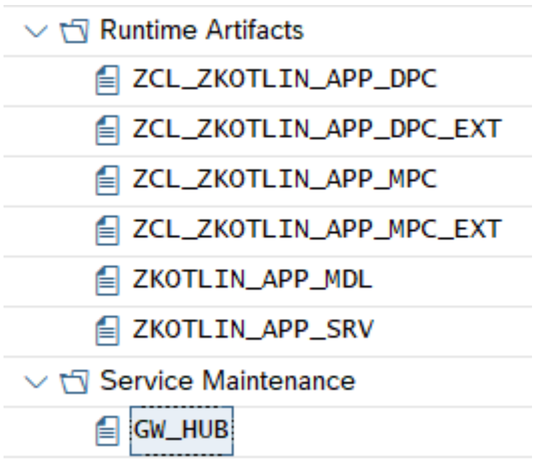
Parameters Exceptions Sourcecode Filter

Method	Level	Visibility	M...	Description
CUSTOMERMEMOSET_GET_ENTITY	Instance Method	Protected		Related EntitySet Name: CustomerMemoSet
CUSTOMERMEMOSET_GET_ENTITYSET	Instance Method	Protected		Related EntitySet Name: CustomerMemoSet

Testing the SAP Gateway

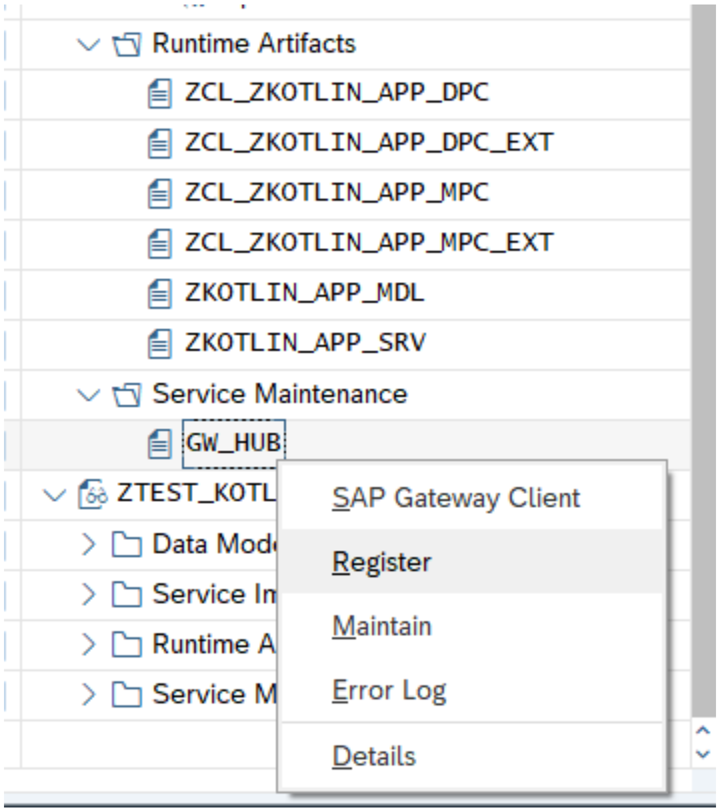
At this point, we have our first 2 methods for retrieving all records or a single record, so let's pause for a moment and walk through how we can test them using the SAP Gateway Client. After all, we don't want to toss this over the wall to our Kotlin developers, without first unit testing it.

Go back to the SAP Gateway Service Builder, and expand the node "Service Maintenance":

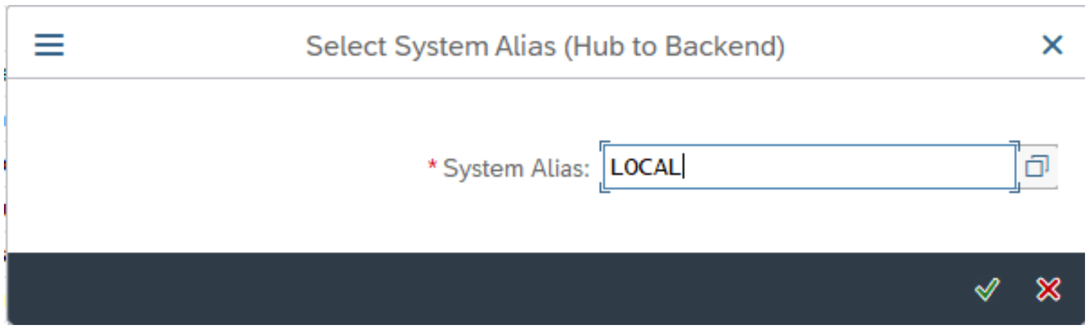

Runtime Artifacts
ZCL_ZKOTLIN_APP_DPC
ZCL_ZKOTLIN_APP_DPC_EXT
ZCL_ZKOTLIN_APP_MPC
ZCL_ZKOTLIN_APP_MPC_EXT
ZKOTLIN_APP_MDL
ZKOTLIN_APP_SRV
Service Maintenance
GW_HUB

We have a Gateway Hub defined as "GW_HUB". If there is nothing under your "Service Maintenance" folder, this means that the SAP Gateway has not yet been setup for your SAP instance. You will need to set this up first. There are many good blogs out there that walk-through a setup.

Right-click on the Gateway defined for your system, in this case GW_HUB and select "Register":


Runtime Artifacts
ZCL_ZKOTLIN_APP_DPC
ZCL_ZKOTLIN_APP_DPC_EXT
ZCL_ZKOTLIN_APP_MPC
ZCL_ZKOTLIN_APP_MPC_EXT
ZKOTLIN_APP_MDL
ZKOTLIN_APP_SRV
Service Maintenance
GW_HUB
ZTEST_KOTL
Data Mod
Service In
Runtime A
Service M

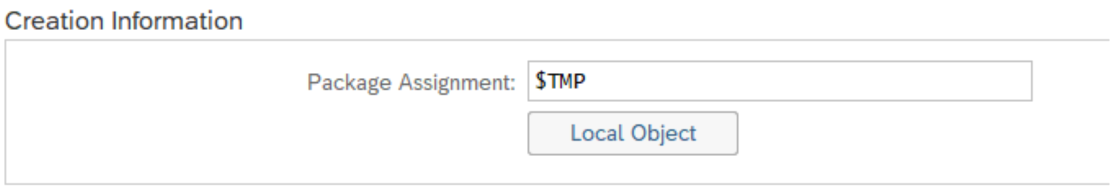
A typical test setup uses the “LOCAL” Alias, which we will use here:



Select System Alias (Hub to Backend)

* System Alias:	LOCAL

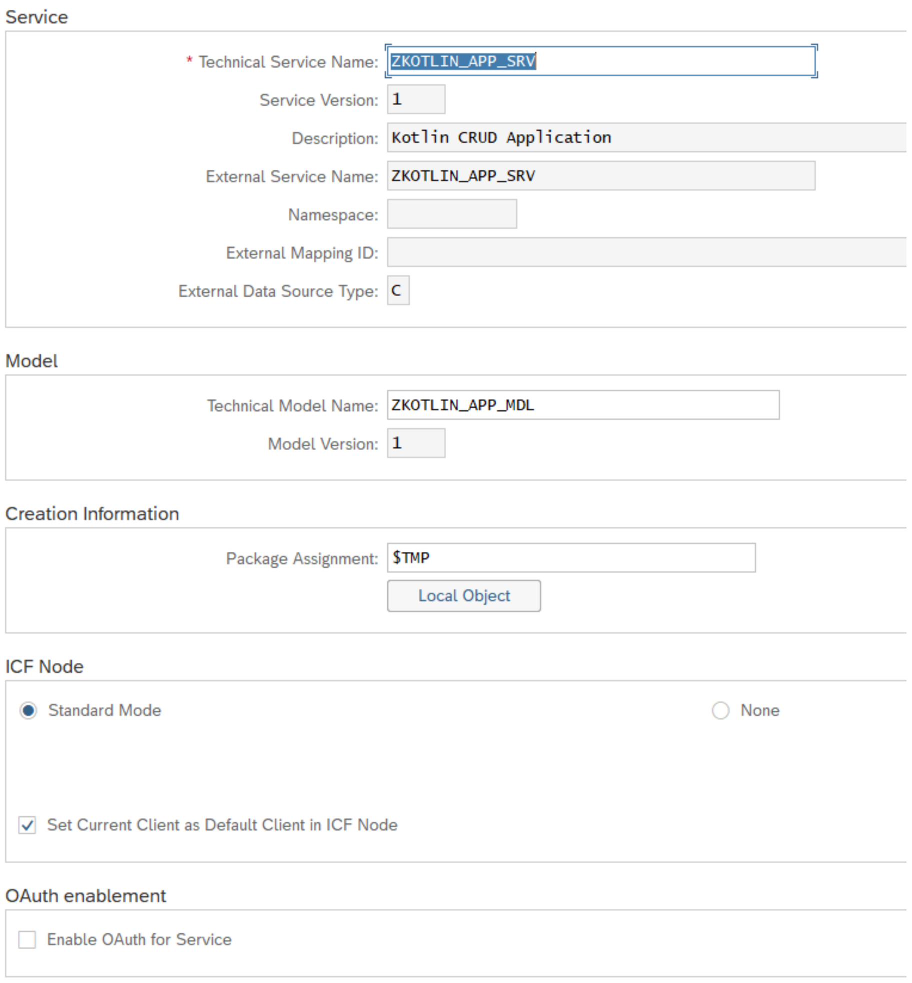
In the center of the next screen, click the button “Local Object”, to avoid creating a transport:



Creation Information

Package Assignment:	$TMP

Hit the Enter key, to accept all of the proposed defaults:



Service

* Technical Service Name:	ZKOTLIN_APP_SRV
Service Version:	1
Description:	Kotlin CRUD Application
External Service Name:	ZKOTLIN_APP_SRV
Namespace:	
External Mapping ID:	
External Data Source Type:	C

Model

Technical Model Name:	ZKOTLIN_APP_MDL
Model Version:	1

Creation Information

Package Assignment:	$TMP

ICF Node

[✓ Standard Mode] None

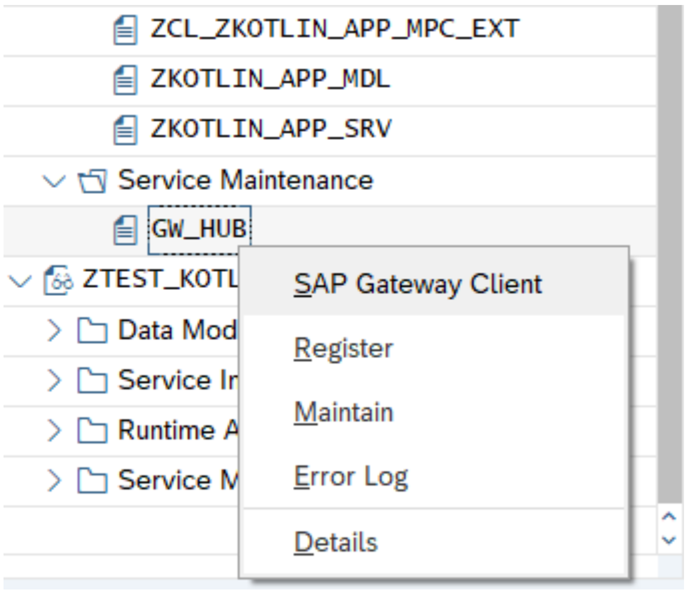
✓ Set Current Client as Default Client in ICF Node

OAuth enablement

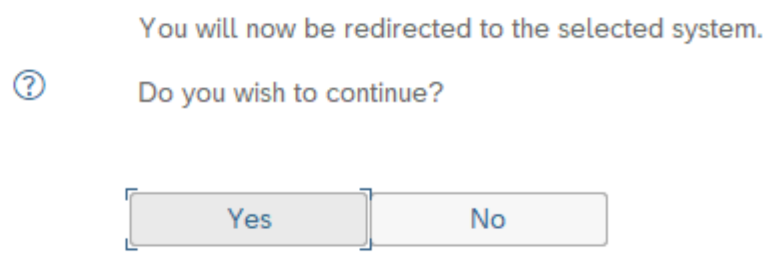
☐ Enable OAuth for Service

\

Back to our first screen, right click on the Gateway hub again, and select “SAP Gateway Client”:

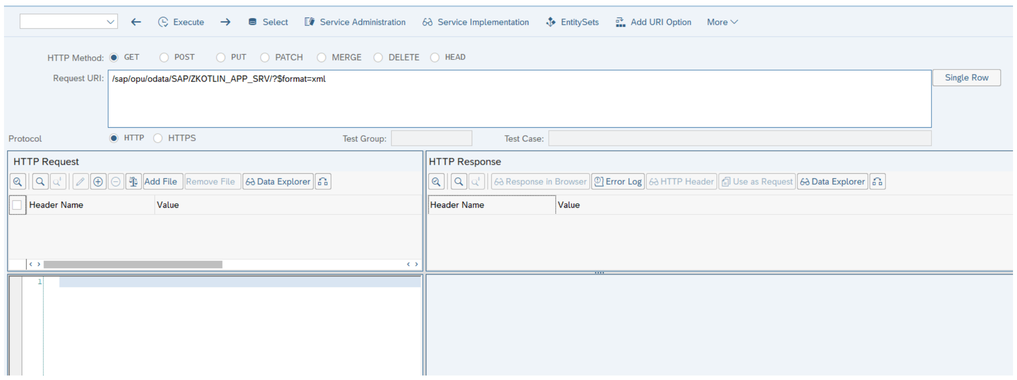


Click “Yes” to be redirected to the system:


You will now be redirected to the selected system.

Do you wish to continue?

This brings us to our SAP Gateway Client, where we may test our new service:



HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

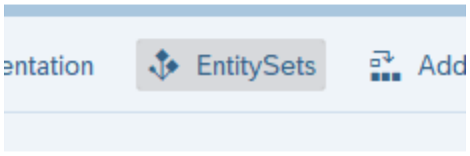
Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/?$format=xml

Protocol [✓ HTTP] HTTPS

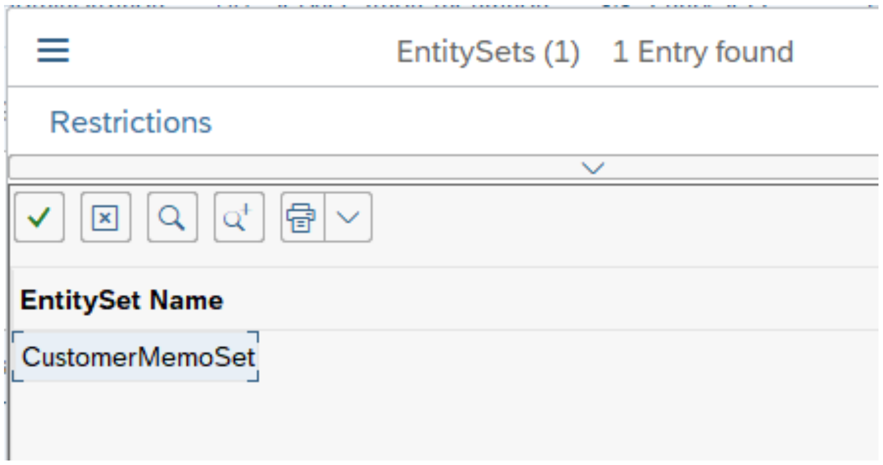
Test Group:

Test Case:

Click on the “EntitySets” button:



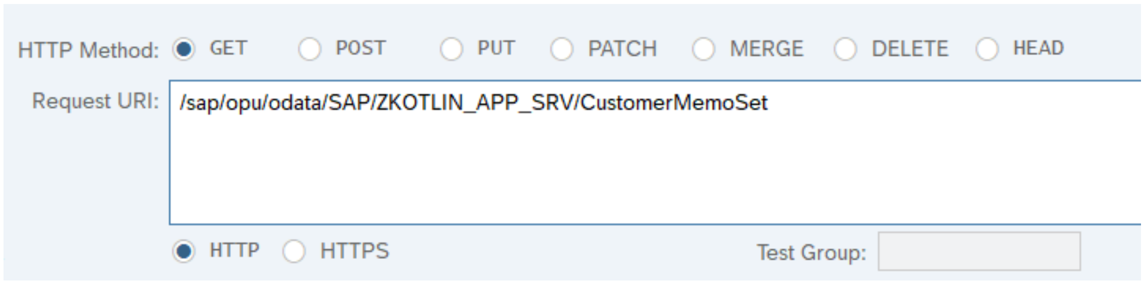
This will bring up our Entity Set that we defined:



EntitySet Name

CustomerMemoSet

Double-click on it, to fetch the URI string for our Entity:

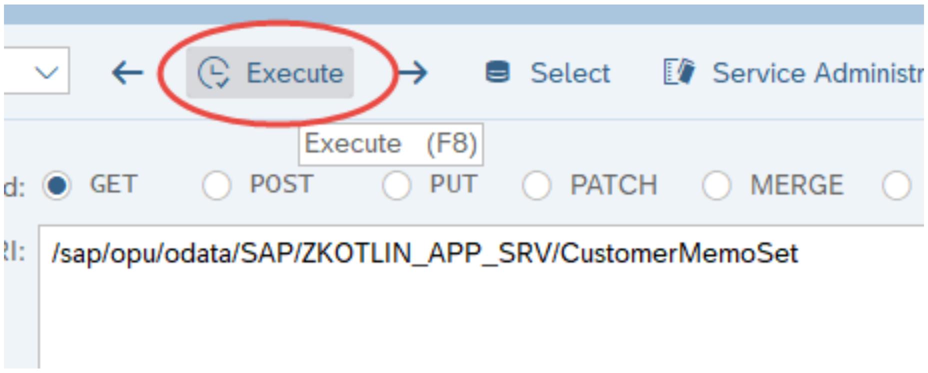


HTTP Method: (● GET) ○ POST ○ PUT ○ PATCH ○ MERGE ○ DELETE ○ HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

(● HTTP) ○ HTTPS Test Group:

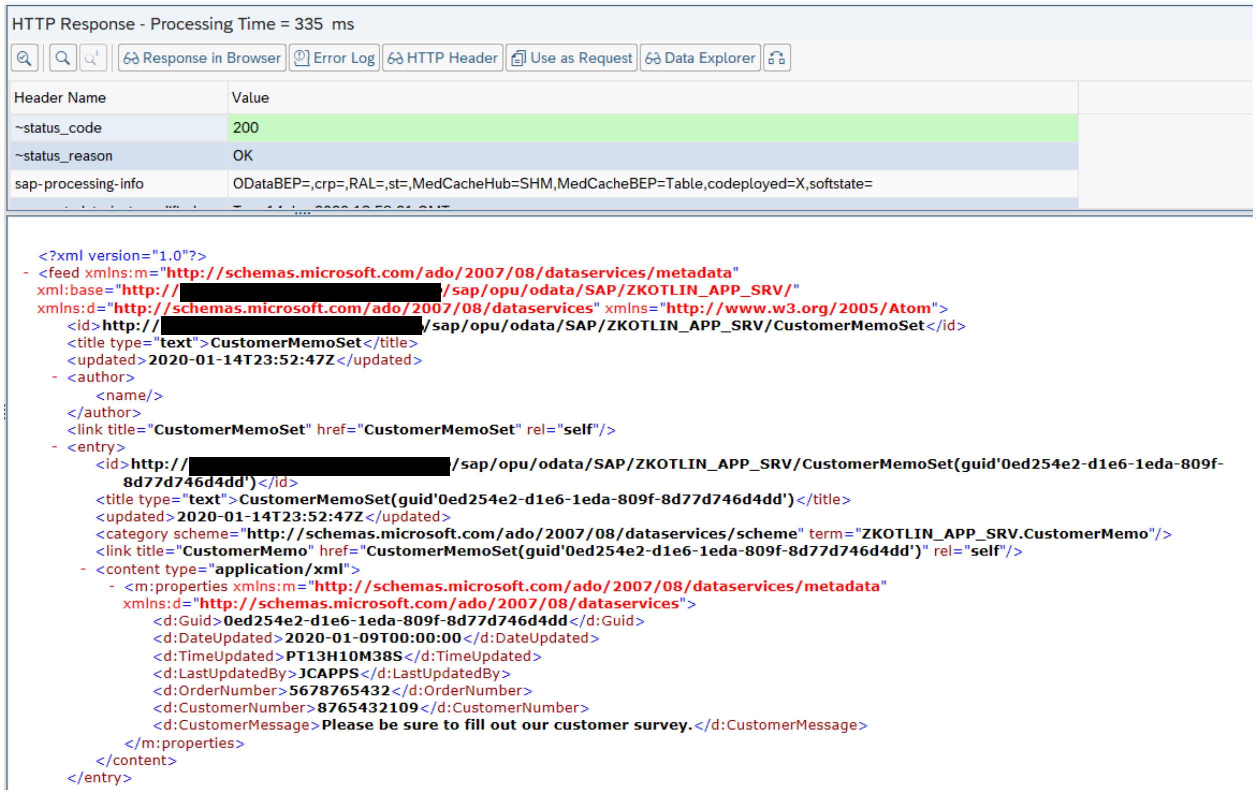
If you previously created test data in the ZTEST_KOTLIN table, you'll get some data. Click on the "Execute" button:



d: (● GET) ○ POST ○ PUT ○ PATCH ○ MERGE ○

I: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

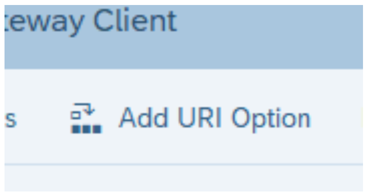
I have 4 records in my table, and all records will be retrieved in OData XML Format on the right-hand pane:



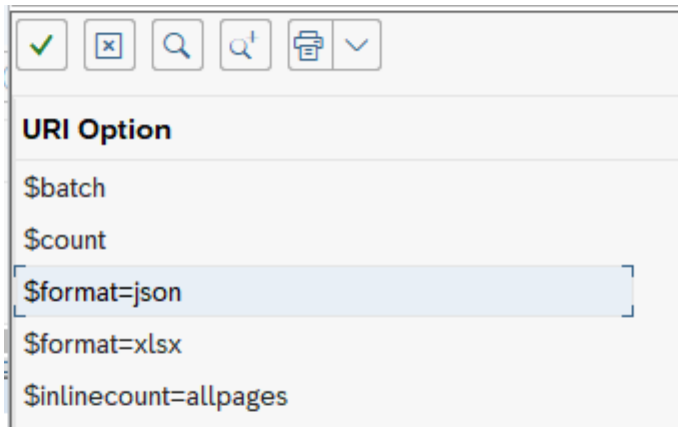
Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,MedCacheBEP=Table,codeployed=X,softstate=

```
 <?xml version="1.0"?>
- <feed xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
 xml:base="http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/"
 xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns="http://www.w3.org/2005/Atom">
     <id>http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet</id>
     <title type="text">CustomerMemoSet</title>
     <updated>2020-01-14T23:52:47Z</updated>
   - <author>
         <name/>
     </author>
     <link title="CustomerMemoSet" href="CustomerMemoSet" rel="self"/>
   - <entry>
         <id>http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-
             8d77d746d4dd')</id>
         <title type="text">CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')</title>
         <updated>2020-01-14T23:52:47Z</updated>
         <category scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" term="ZKOTLIN_APP_SRV.CustomerMemo"/>
         <link title="CustomerMemo" href="CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')" rel="self"/>
       - <content type="application/xml">
           - <m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
             xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
                 <d:Guid>0ed254e2-d1e6-1eda-809f-8d77d746d4dd</d:Guid>
                 <d:DateUpdated>2020-01-09T00:00:00</d:DateUpdated>
                 <d:TimeUpdated>PT13H10M38S</d:TimeUpdated>
                 <d:LastUpdatedBy>JCAPPS</d:LastUpdatedBy>
                 <d:OrderNumber>5678765432</d:OrderNumber>
                 <d:CustomerNumber>8765432109</d:CustomerNumber>
                 <d:CustomerMessage>Please be sure to fill out our customer survey.</d:CustomerMessage>
             </m:properties>
         </content>
     </entry>
```

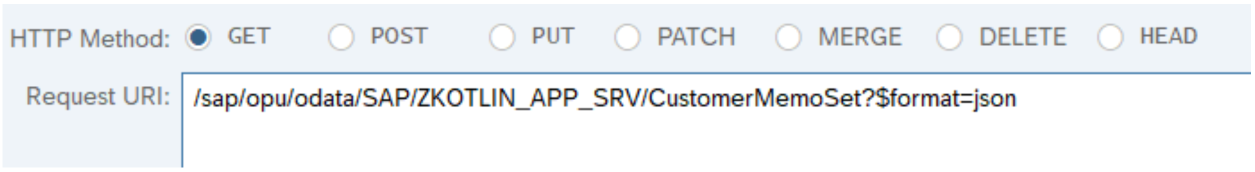
Click on the button “Add URI Option”:



Double-click on “$format=json”:

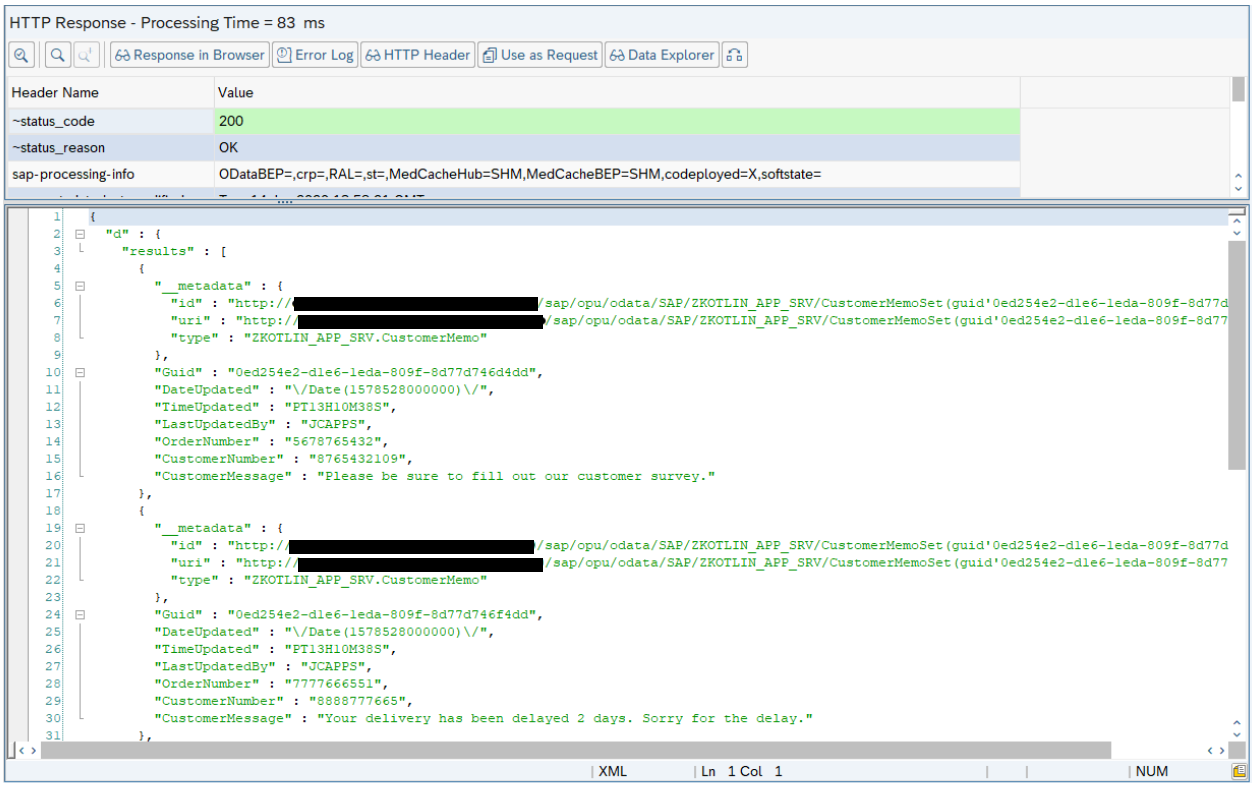


This will append a JSON uri option to the end of our request:


HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet?$format=json

Execute again, and our data is now in JSON Format (rather than standard default OData xml):

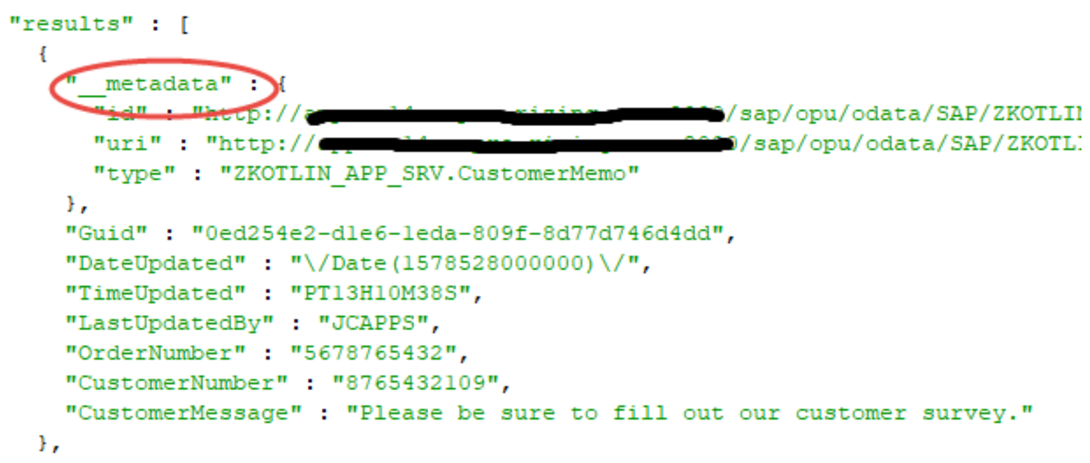

HTTP Response - Processing Time = 83 ms

Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,MedCacheBEP=SHM,codeployed=X,softstate=

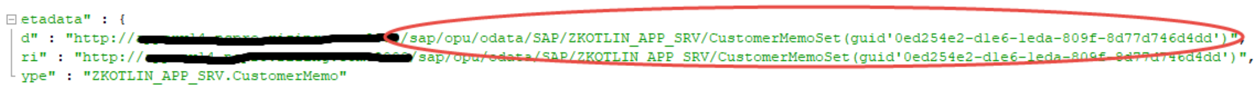
```
{
  "d" : {
    "results" : [
      {
        "__metadata" : {
          "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-dle6-leda-809f-8d77d
          "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-dle6-leda-809f-8d77
          "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
        },
        "Guid" : "0ed254e2-dle6-leda-809f-8d77d746d4dd",
        "DateUpdated" : "\/Date(1578528000000)\/",
        "TimeUpdated" : "PT13H10M38S",
        "LastUpdatedBy" : "JCAPPS",
        "OrderNumber" : "5678765432",
        "CustomerNumber" : "8765432109",
        "CustomerMessage" : "Please be sure to fill out our customer survey."
      },
      {
        "__metadata" : {
          "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-dle6-leda-809f-8d77d
          "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-dle6-leda-809f-8d77
          "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
        },
        "Guid" : "0ed254e2-dle6-leda-809f-8d77d746f4dd",
        "DateUpdated" : "\/Date(1578528000000)\/",
        "TimeUpdated" : "PT13H10M38S",
        "LastUpdatedBy" : "JCAPPS",
        "OrderNumber" : "7777666551",
        "CustomerNumber" : "8888777665",
        "CustomerMessage" : "Your delivery has been delayed 2 days. Sorry for the delay."
      },
```

For our Kotlin application, because JSON is the most Java and Kotlin friendly format, we'll be fetching our data in JSON format by appending this URI option.

Next, let's test the GET for a single Record. Notice when we fetch all records, each record has it's own "__metadata" attribute:


```
"results" : [
  {
    "__metadata" : {
      "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLII
      "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTL
      "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
    },
    "Guid" : "0ed254e2-dle6-leda-809f-8d77d746d4dd",
    "DateUpdated" : "\/Date(1578528000000)\/",
    "TimeUpdated" : "PT13H10M38S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "5678765432",
    "CustomerNumber" : "8765432109",
    "CustomerMessage" : "Please be sure to fill out our customer survey."
  },
```

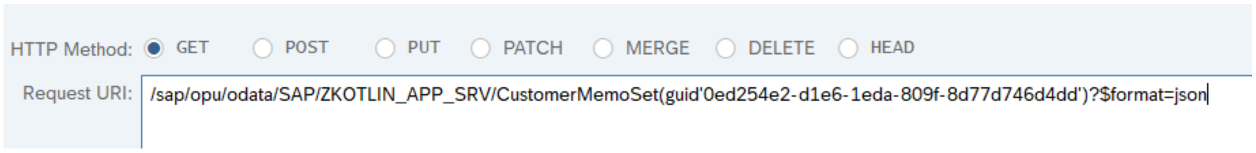
We can get the URI for that individual record, by copying that string, starting with "/sap/opu/":



```
etadata" : {
d" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')",
ri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')",
ype" : "ZKOTLIN_APP_SRV.CustomerMemo"
```

Paste the following string into the Request URI, to fetch a single record:

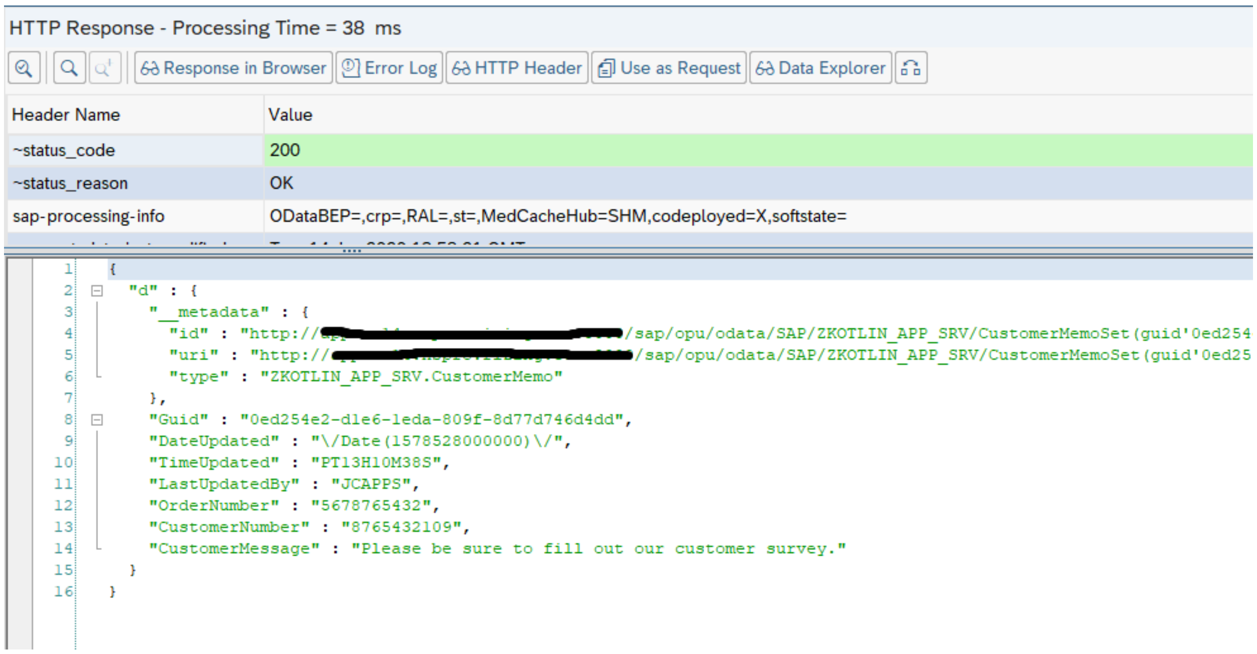
/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')?$format=json



HTTP Method: ● GET ○ POST ○ PUT ○ PATCH ○ MERGE ○ DELETE ○ HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')?$format=json

Execute, and the results will contain a single record:



HTTP Response - Processing Time = 38 ms

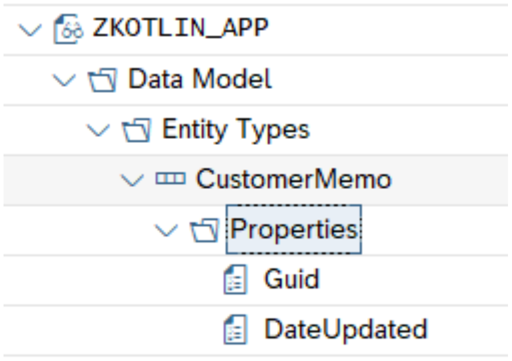
Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=

```
{
  "d" : {
    "__metadata" : {
      "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254
      "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed25
      "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
    },
    "Guid" : "0ed254e2-d1e6-1eda-809f-8d77d746d4dd",
    "DateUpdated" : "\/Date(1578528000000)\/",
    "TimeUpdated" : "PT13H10M38S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "5678765432",
    "CustomerNumber" : "8765432109",
    "CustomerMessage" : "Please be sure to fill out our customer survey."
  }
}
```

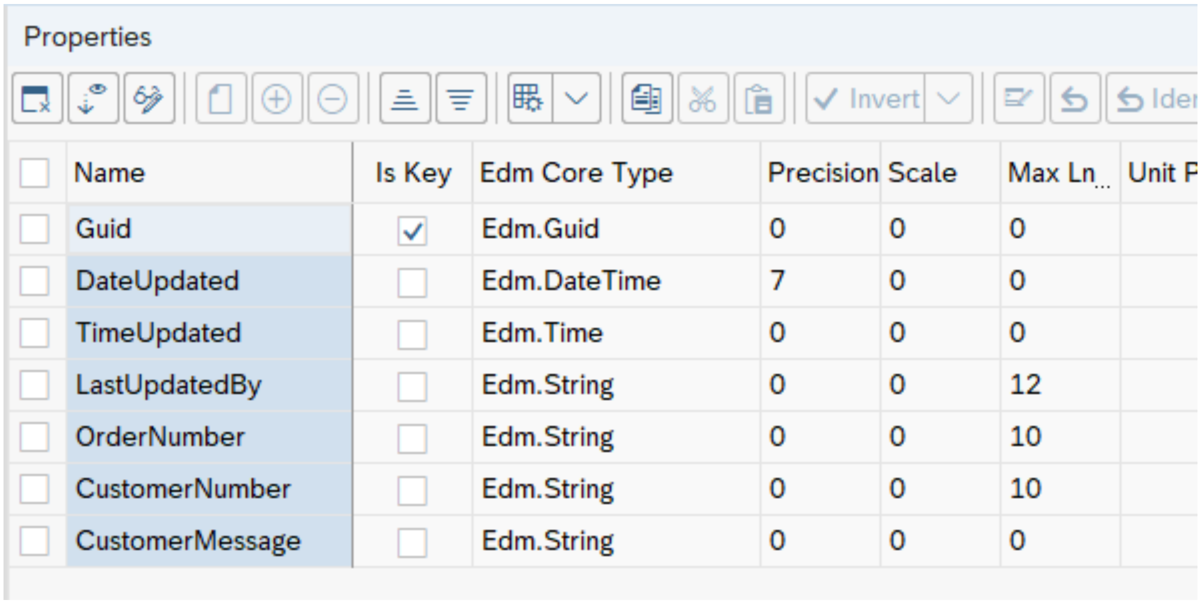
One thing I wanted to point out, which you could get stuck on. Typically, if you are selecting a single record, and the key is a string, you can simply specify it. For example:

```
VendorGetSet/?$filter=(Matnr eq '10000030')
```

If you tried to say (Guid eq '<your guid>'), you would get an error. If we go back to our Gateway Project, and double-click on "Properties" for our CustomerMemo Entity:



Properties:



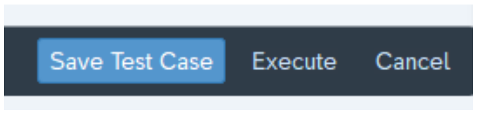
	Name	Is Key	Edm Core Type	Precision	Scale	Max Ln...	Unit P
☐	Guid	✓	Edm.Guid	0	0	0	
☐	DateUpdated	☐	Edm.DateTime	7	0	0	
☐	TimeUpdated	☐	Edm.Time	0	0	0	
☐	LastUpdatedBy	☐	Edm.String	0	0	12	
☐	OrderNumber	☐	Edm.String	0	0	10	
☐	CustomerNumber	☐	Edm.String	0	0	10	
☐	CustomerMessage	☐	Edm.String	0	0	0	

Notice our key field Guid, is of type "Edm.Guid":

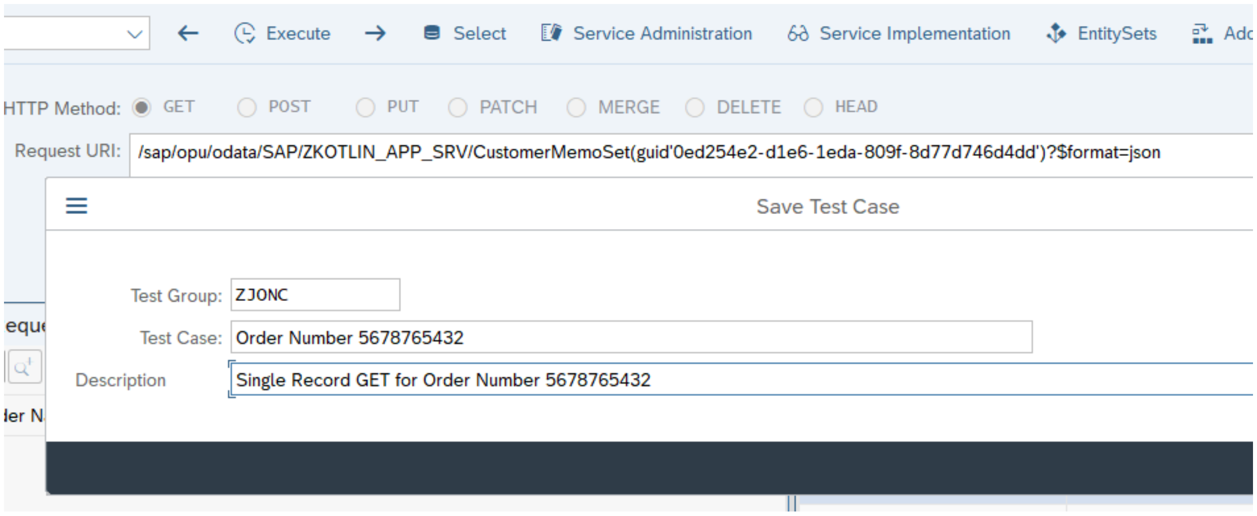
`Edm.Guid`	`A 16-byte (128-bit) unique identifier value`
`Edm.Int16`	`A signed 16-bit integer value`

The internal ABAP Type is TYPEKIND_HEX. When specifying this data type, you must place "guid" within the GET string – "CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')".

One other feature of the SAP Gateway Client is the ability to save our test cases. For example, if we'd like to save this GET test case, which retrieves this single record, we can go to the lower-right of the screen and click "Save Test Case":



We can also setup Test Groups:



HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')?$format=json

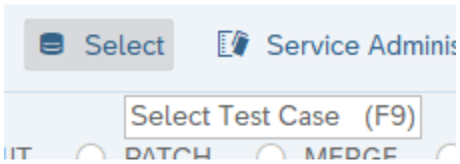
Save Test Case

Test Group: ZJONC

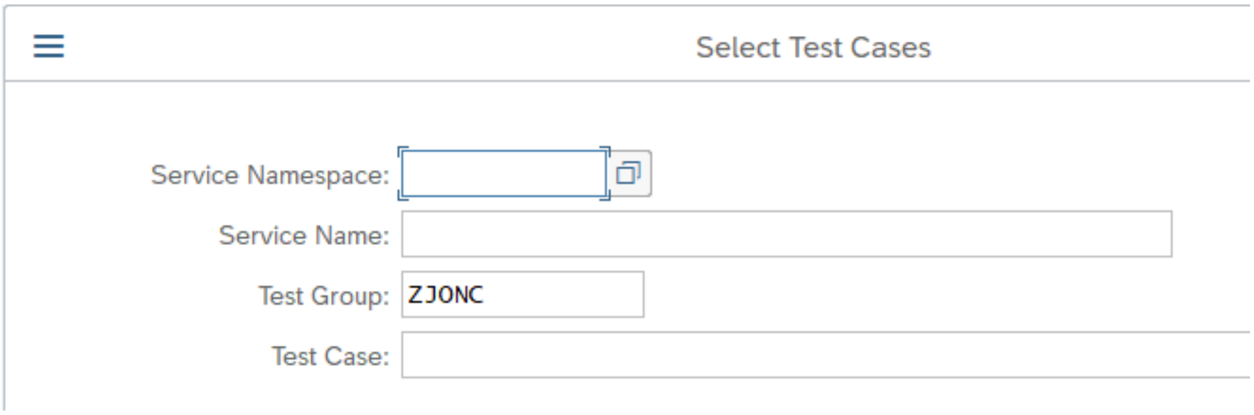
Test Case: Order Number 5678765432

Description: Single Record GET for Order Number 5678765432

Now, we can retrieve this test case anytime, by clicking on the "Select" button:



Enter my Test Group:


Select Test Cases

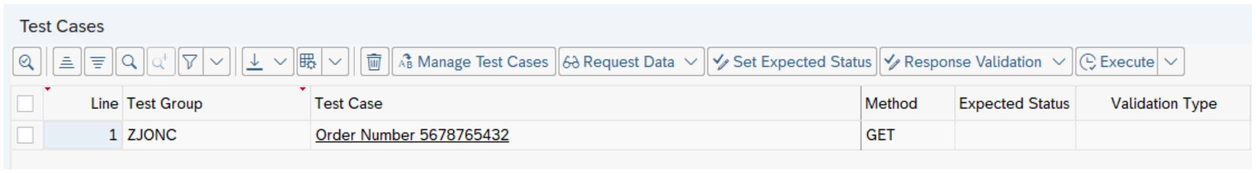
Service Namespace:

Service Name:

Test Group: ZJONC

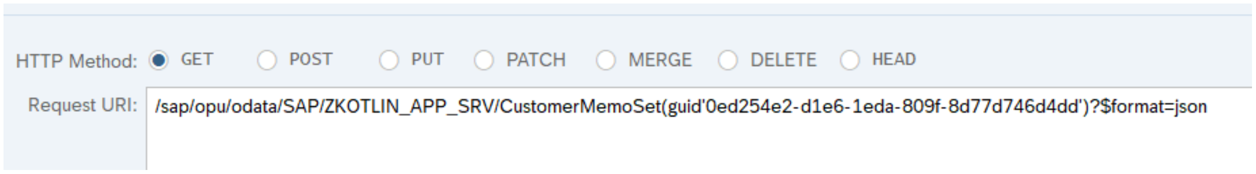
Test Case:

Lists all tests in this test group:


Test Cases

Line	Test Group	Test Case	Method	Expected Status	Validation Type
1	ZJONC	Order Number 5678765432	GET		

Clicking on the “Test Case” link, will auto-populate the URI, with your previous test case:


HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

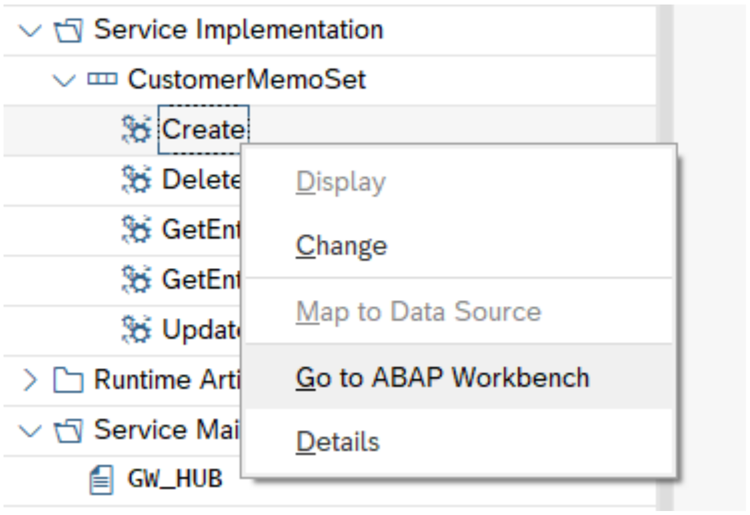
Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')?$format=json

You can now re-execute this test.

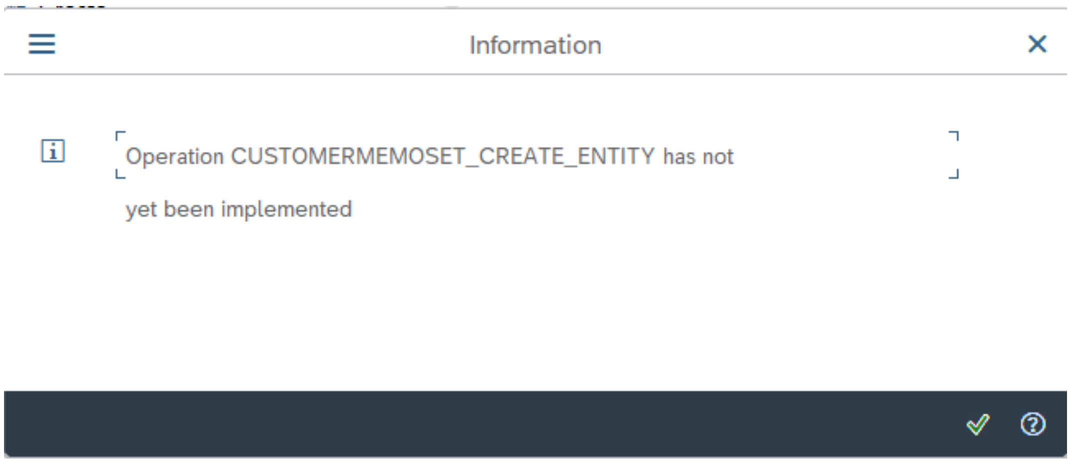
Testing a “GET” is pretty straightforward. Testing is a little trickier for create and update operations, so each section, below, includes testing instructions and how to include a payload with the HTTP Request. See the below section “Create Record” for the most detailed testing instructions on testing with a payload, and some recommended testing tools.

CREATE RECORD

To enable the HTTP POST method, right-click on the “Create” operation under the “Service Implementation” node and click “Go to ABAP Workbench”:

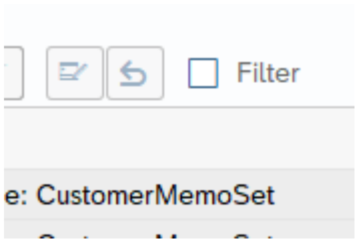


You’ll get a message that this operation has not yet been implemented, hit Enter to continue:


Information

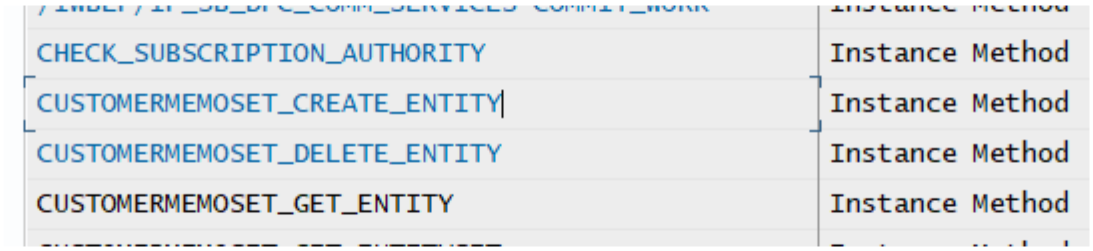
Operation CUSTOMERMEMOSET_CREATE_ENTITY has not yet been implemented

Ensure the “Filter” on the upper right is unchecked, so we can find our CUSTOMERMEMOSET_CREATE_ENTITY method:



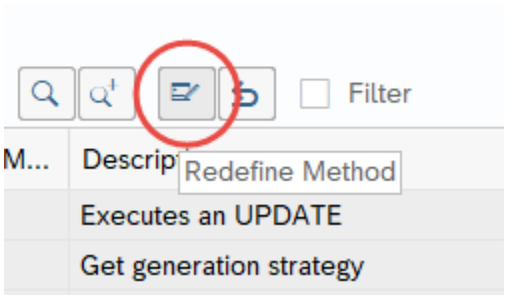
Switch to Change Mode to make the class editable.

Click on this method, to place your cursor into the row for method CUSTOMERMEMOSET_CREATE_ENTITY:

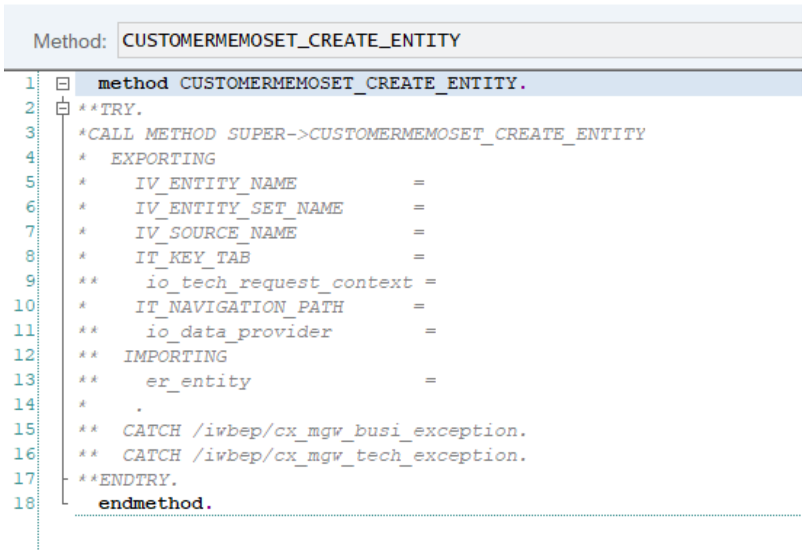


CHECK_SUBSCRIPTION_AUTHORITY	Instance Method
CUSTOMERMEMOSET_CREATE_ENTITY	Instance Method
CUSTOMERMEMOSET_DELETE_ENTITY	Instance Method
CUSTOMERMEMOSET_GET_ENTITY	Instance Method

Next, click on the “Redefine Method” button:



This brings you into the code for this method:



Method: CUSTOMERMEMOSET_CREATE_ENTITY

```
  method CUSTOMERMEMOSET_CREATE_ENTITY.
**TRY.
*CALL METHOD SUPER->CUSTOMERMEMOSET_CREATE_ENTITY
*  EXPORTING
*    IV_ENTITY_NAME          =
*    IV_ENTITY_SET_NAME      =
*    IV_SOURCE_NAME          =
*    IT_KEY_TAB              =
**    io_tech_request_context =
*    IT_NAVIGATION_PATH      =
**    io_data_provider        =
**  IMPORTING
**    er_entity               =
*    .
**  CATCH /iwbep/cx_mgw_busi_exception.
**  CATCH /iwbep/cx_mgw_tech_exception.
**ENDTRY.
  endmethod.
```

Replace the above commented code with the following:

```
  METHOD customermemoset_create_entity.
    DATA: lw_record TYPE zcl_zkotlin_app_mpc=>ts_customermemo.

    io_data_provider->read_entry_data( IMPORTING es_data = lw_record ).
    IF lw_record-order_number     IS INITIAL OR
       lw_record-customer_number  IS INITIAL OR
       lw_record-customer_message IS INITIAL.
      RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception.
    ENDIF.

    "Create a record in the database...
    CLEAR: lw_record-date_updated, lw_record-time_updated, lw_record-last_updated_by,
           lw_record-guid.
    lw_record-date_updated    = sy-datum.
    lw_record-time_updated    = sy-uzeit.
    lw_record-last_updated_by = sy-uname.
    "Generate a unique key (guid)...
    TRY.
        lw_record-guid = cl_system_uuid=>if_system_uuid_static~create_uuid_x16( ).
      CATCH cx_uuid_error INTO DATA(lo_guid_error).
        RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception.
    ENDTRY.

    INSERT ztest_kotlin FROM lw_record.
    COMMIT WORK AND WAIT.
    IF sy-subrc = 0.
      "New record was created successfully...
      er_entity = lw_record.
    ELSE.
      "Entity not found
      RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
        EXPORTING
          textid      = /iwbep/cx_mgw_busi_exception=>resource_not_found
          entity_type = iv_entity_name.
    ENDIF.
  ENDMETHOD.
```

Save and Activate the above code.

The above code does the following:

Fetch the incoming record from the Gateway:

```
io_data_provider->read_entry_data( IMPORTING es_data = lw_record ).
```

For a creation, the fields we need from the user are order number, customer number and customer message (i.e. Memo). Aside from that, we will auto-populate all other fields:

date_updated = Date the user created or last updated the record.
time_updated = Time the user created or last updated the record.
last_updated_by = The SAP user name of the user who last created or updated the record.

SAP provides a class to auto-generate a unique GUID, which will represent the new key for our new record:

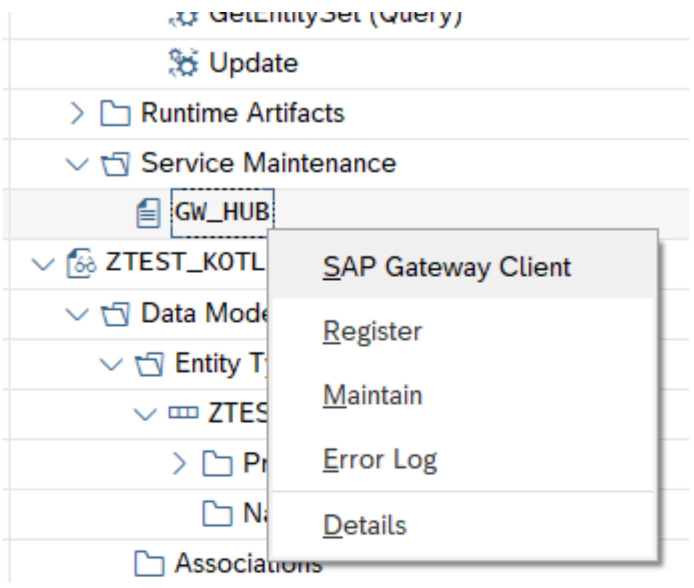
```
"Generate a unique key (guid)...
TRY.
    lw_record-guid = cl_system_uuid=>if_system_uuid_static~create_uuid_x16( ).
  CATCH cx_uuid_error INTO DATA(lo_guid_error).
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception.
ENDTRY.
```

Finally, we do the insert and commit into the database.

Now, to test takes a bit of effort. You can easily waste a couple of days figuring this one out, so I hope the following steps will save you some time!

When testing with the SAP Gateway Client, some authorization items are handled for you, so keep this in mind. Let's first test with the SAP Gateway Client, to start with the easiest approach.

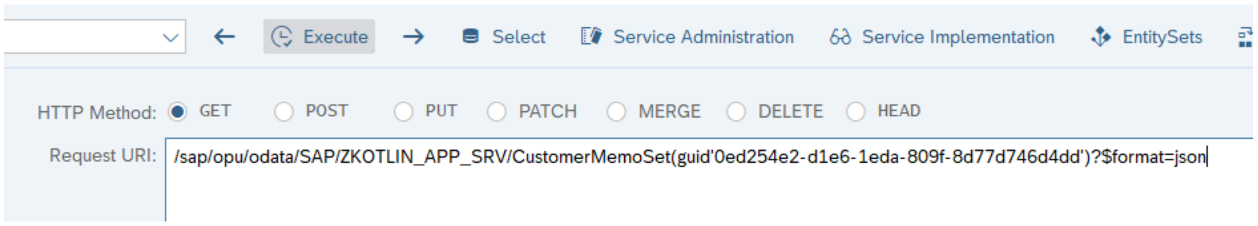
Go back to our project, and right click on your Gateway hub, then "SAP Gateway Client":



There is a feature, whereas you can perform an HTTP GET, then turn this data into an update method (i.e. POST, PUT, DELETE, etc.). This is the easiest way to get the properly formatted data, etc. Let's use our URI from earlier, to get a single record. Enter the following into the Request URI or click "Select" and retrieve the earlier test case:

```
/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-
8d77d746d4dd')?$format=json
```

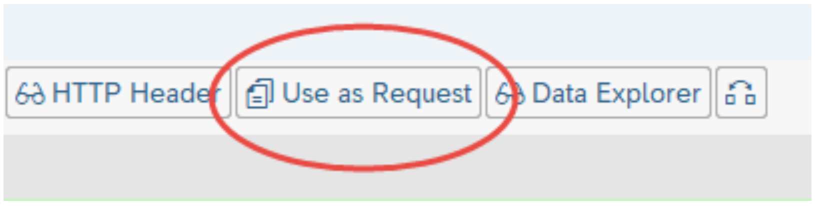
Now, execute the above:



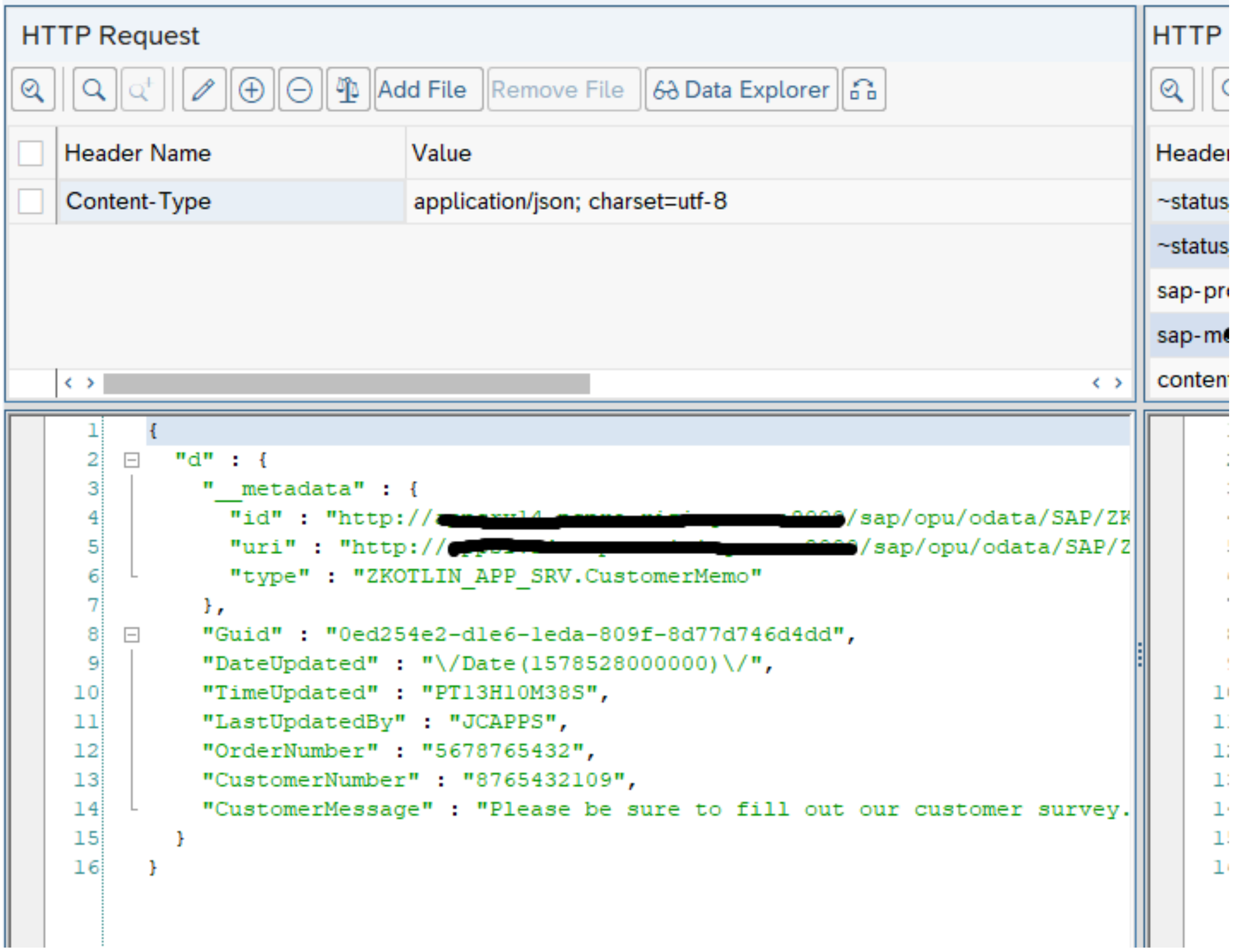
HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d746d4dd')?$format=json

On the top of the HTTP Response section, click on the "Use as Request" button:



This moves the data into the left-hand pane for an HTTP Request:

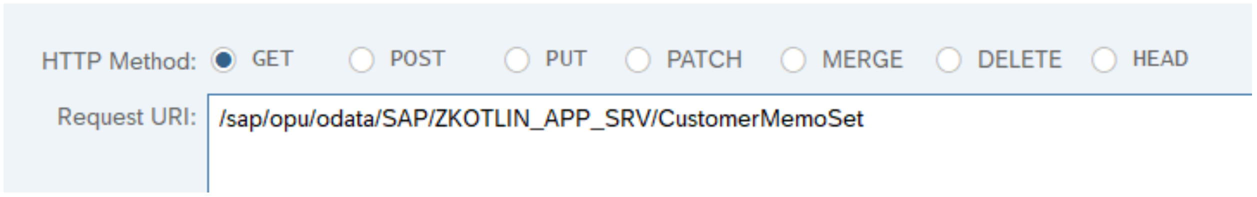


Header Name	Value
Content-Type	application/json; charset=utf-8

```
{
  "d" : {
    "__metadata" : {
      "id" : "http://[redacted]/sap/opu/odata/SAP/ZK
      "uri" : "http://[redacted]/sap/opu/odata/SAP/Z
      "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
    },
    "Guid" : "0ed254e2-d1e6-1eda-809f-8d77d746d4dd",
    "DateUpdated" : "\/Date(1578528000000)\/",
    "TimeUpdated" : "PT13H10M38S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "5678765432",
    "CustomerNumber" : "8765432109",
    "CustomerMessage" : "Please be sure to fill out our customer survey.
  }
}
```

Next, remove the JSON URI Option and the GUID at the end of the string. You now have the following URI:

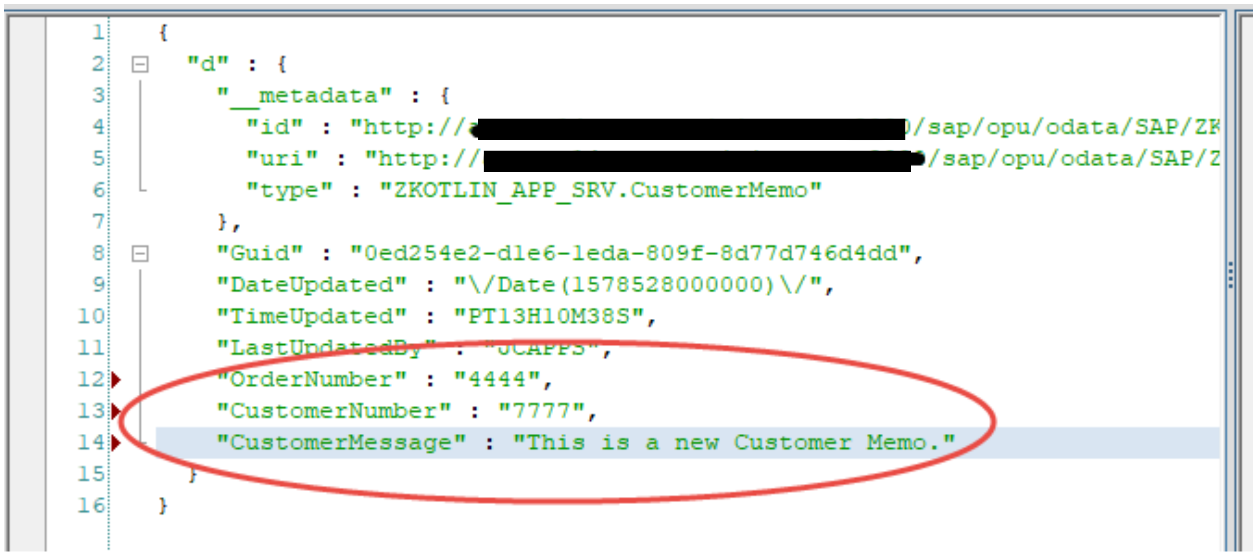
`/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet`



HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

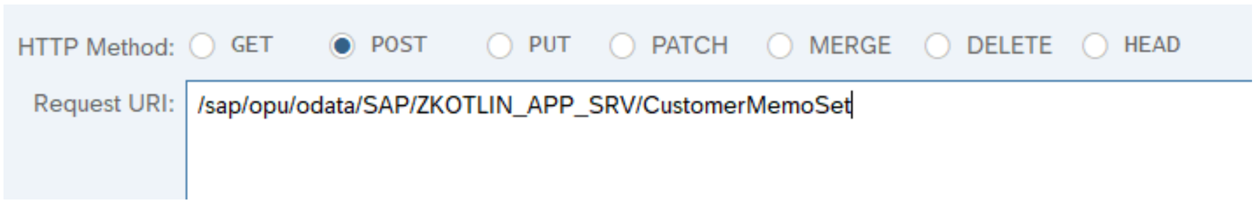
In looking at our custom code in the method, we recall that the required fields are Order Number, Customer Number and Customer Message (Memo). Let's overwrite those fields in the HTTP Request, with our new data:



```
{
  "d" : {
    "__metadata" : {
      "id" : "http://[redacted]/sap/opu/odata/SAP/ZK
      "uri" : "http://[redacted]/sap/opu/odata/SAP/Z
      "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
    },
    "Guid" : "0ed254e2-d1e6-1eda-809f-8d77d746d4dd",
    "DateUpdated" : "\/Date(1578528000000)\/",
    "TimeUpdated" : "PT13H10M38S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "4444",
    "CustomerNumber" : "7777",
    "CustomerMessage" : "This is a new Customer Memo."
  }
}
```

Even though we are doing a "Create", you still need to specify a GUID and all fields in the payload request. The GUID could be all zeros but must be in the same format. Since our method will generate a GUID and ignore the one that is passed in, we'll just leave ours as-is in the request.

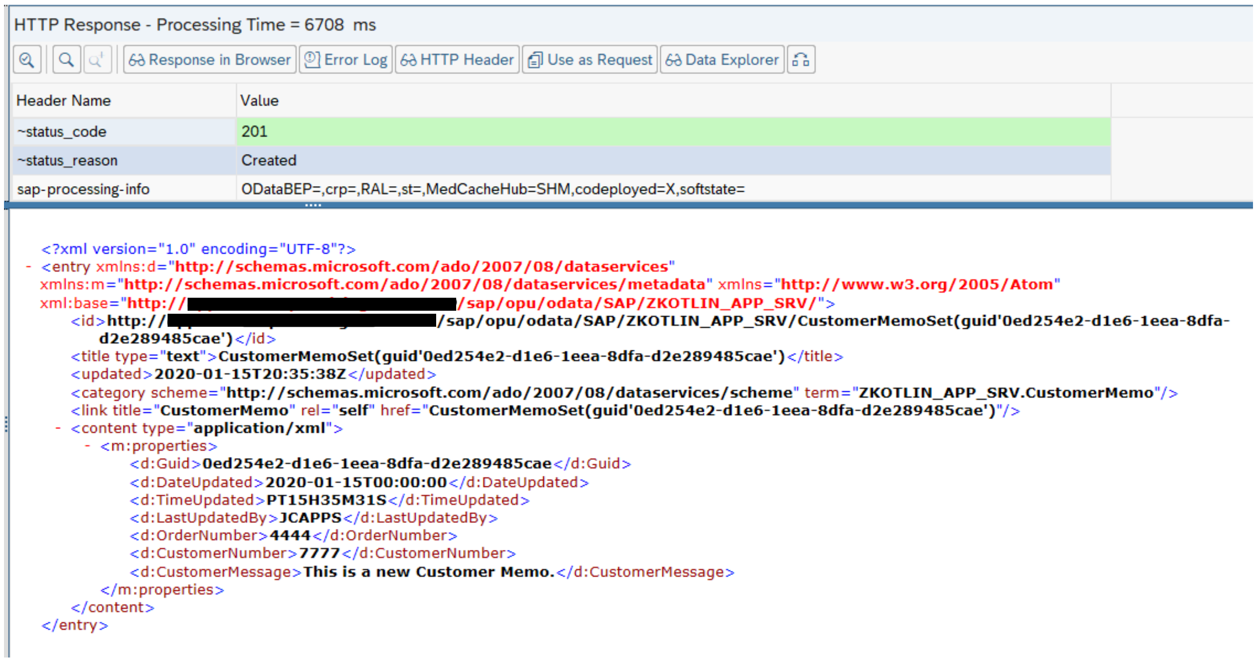
Next, change the HTTP Method to "POST", instead of "GET":



HTTP Method: GET [✓ POST] PUT PATCH MERGE DELETE HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

Now execute and you should see the following, with a success code 201, plus your newly generated data in the response:

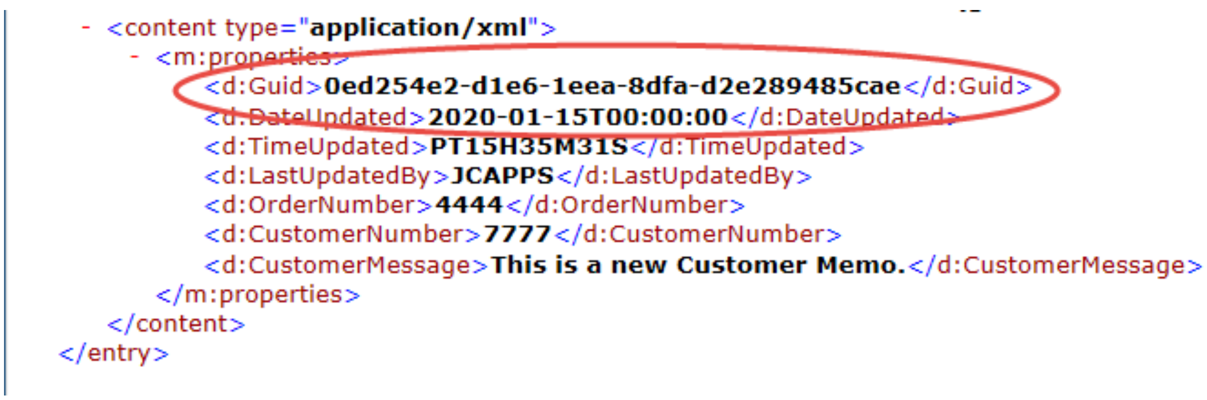


HTTP Response - Processing Time = 6708 ms

Header Name	Value
~status_code	201
~status_reason	Created
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=

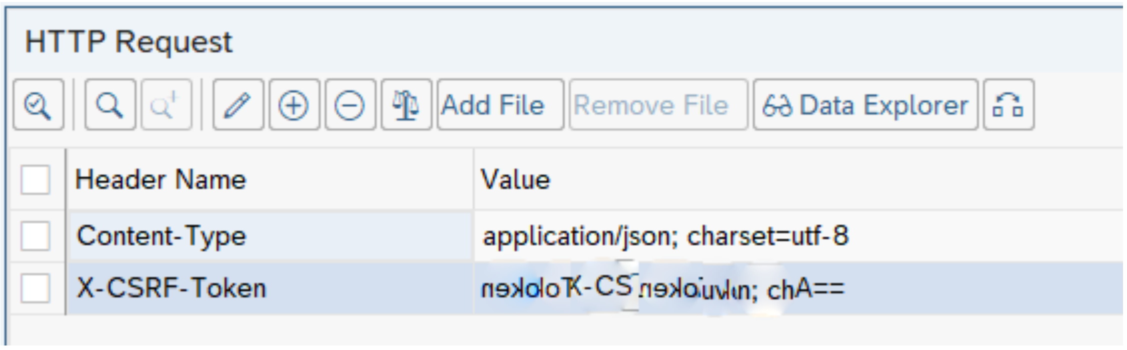
```
  <?xml version="1.0" encoding="UTF-8"?>
- <entry xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2005/Atom"
  xml:base="http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/">
      <id>http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8dfa-
        d2e289485cae')</id>
      <title type="text">CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8dfa-d2e289485cae')</title>
      <updated>2020-01-15T20:35:38Z</updated>
      <category scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" term="ZKOTLIN_APP_SRV.CustomerMemo"/>
      <link title="CustomerMemo" rel="self" href="CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8dfa-d2e289485cae')"/>
    - <content type="application/xml">
        - <m:properties>
              <d:Guid>0ed254e2-d1e6-1eea-8dfa-d2e289485cae</d:Guid>
              <d:DateUpdated>2020-01-15T00:00:00</d:DateUpdated>
              <d:TimeUpdated>PT15H35M31S</d:TimeUpdated>
              <d:LastUpdatedBy>JCAPPS</d:LastUpdatedBy>
              <d:OrderNumber>4444</d:OrderNumber>
              <d:CustomerNumber>7777</d:CustomerNumber>
              <d:CustomerMessage>This is a new Customer Memo.</d:CustomerMessage>
          </m:properties>
      </content>
  </entry>
```

Notice our new record has a new GUID of “0ed254e2-d1e6-1eea-8dfa-d2e289485cae”:



```
    - <content type="application/xml">
        - <m:properties>
              <d:Guid>0ed254e2-d1e6-1eea-8dfa-d2e289485cae</d:Guid>
              <d:DateUpdated>2020-01-15T00:00:00</d:DateUpdated>
              <d:TimeUpdated>PT15H35M31S</d:TimeUpdated>
              <d:LastUpdatedBy>JCAPPS</d:LastUpdatedBy>
              <d:OrderNumber>4444</d:OrderNumber>
              <d:CustomerNumber>7777</d:CustomerNumber>
              <d:CustomerMessage>This is a new Customer Memo.</d:CustomerMessage>
          </m:properties>
      </content>
  </entry>
```

Notice when you execute the POST, a X-CSRF-Token appears in the upper left “HTTP Request” section:

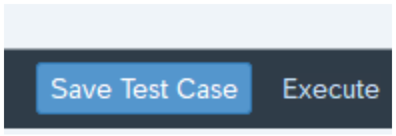


HTTP Request

Header Name	Value
Content-Type	application/json; charset=utf-8
X-CSRF-Token	[illegible]; chA==

The SAP Gateway handles some steps for us, but when we call the service from an external application or testing tool, we’ll need to handle this ourselves. More on this later.

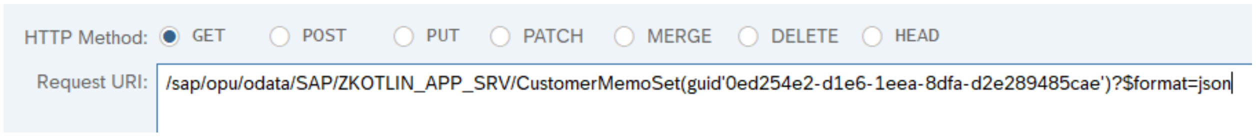
At this point, you may want to save this as a test case for later use, by clicking on the "Save Test Case" button in the lower right corner.



To prove that this worked, copy your newly generated GUID, and do a GET in the test tool, with the URI:

/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8dfa-d2e289485cae')?$format=json

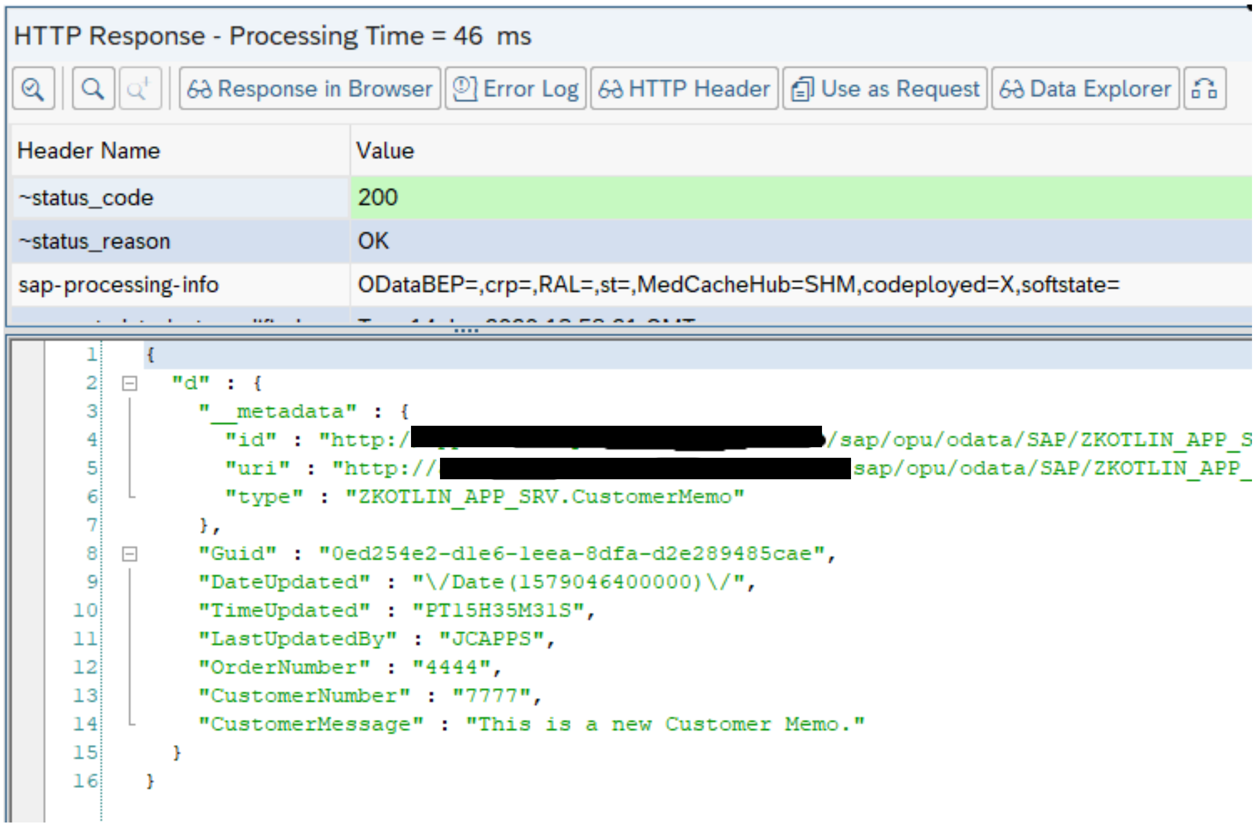
Switch the HTTP Method back to "GET" and execute:



HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8dfa-d2e289485cae')?$format=json

This will fetch our new record:



Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=

```
{
  "d" : {
    "__metadata" : {
      "id" : "http:/[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_S
      "uri" : "http://[redacted]sap/opu/odata/SAP/ZKOTLIN_APP_
      "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
    },
    "Guid" : "0ed254e2-d1e6-1eea-8dfa-d2e289485cae",
    "DateUpdated" : "\/Date(1579046400000)\/",
    "TimeUpdated" : "PT15H35M31S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "4444",
    "CustomerNumber" : "7777",
    "CustomerMessage" : "This is a new Customer Memo."
  }
}
```

Because we will be building a Kotlin application which will be calling the Gateway Service externally, now is a good time to test with another tool, unrelated to the SAP Gateway Client.

2 tools that you could use to test RESTful web services:

Postman:

https://www.getpostman.com/

Advanced REST Client:

https://install.advancedrestclient.com/install

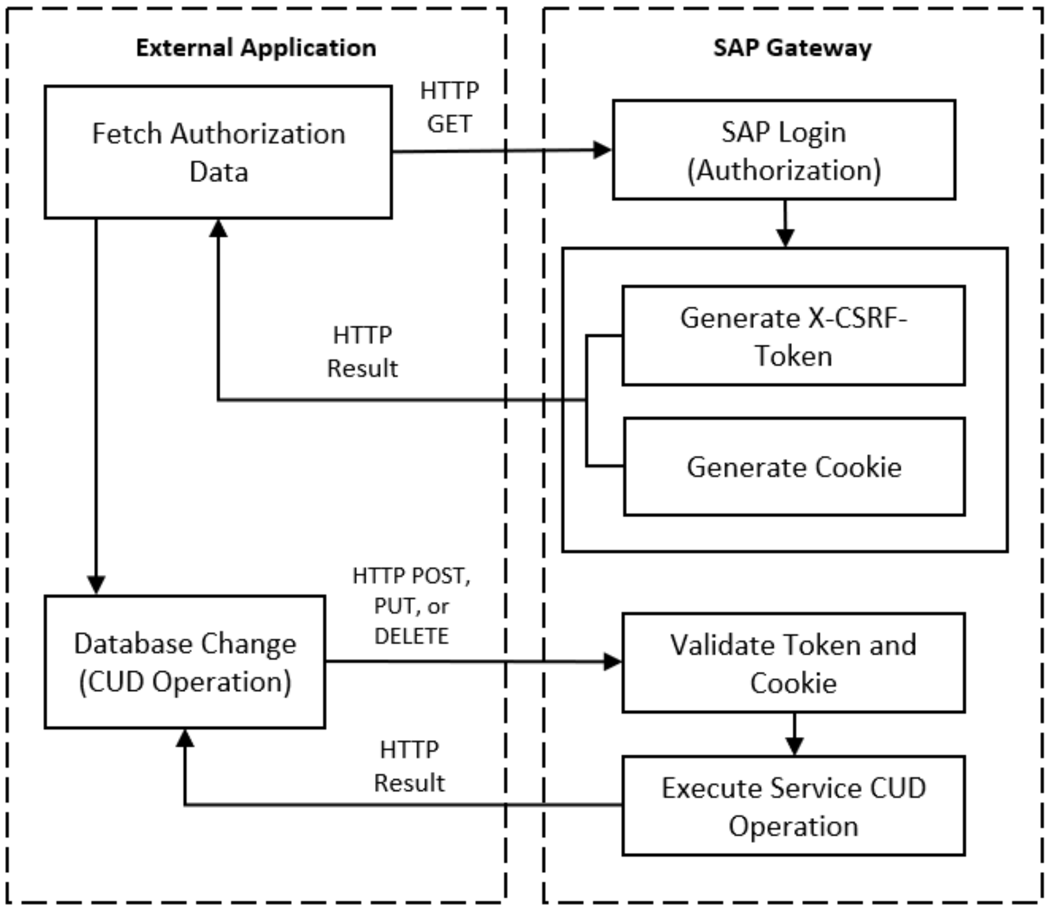
Let's walkthrough a test with the Advanced Rest Client (ARC). There are 3 important items that the SAP Gateway handles for us, which we must do ourselves if calling the service from an external application:

Authorization – This is the same as your SAP Login ID and password. Since we are already logged into SAP when we use the SAP Gateway Client, nothing was needed.

X-CSRF-Token – Cross-Site Request Forgery to prevent malicious attacks on your website or service.

Set Cookie – The SAP Gateway can also generate a cookie, which you can use to call-back the service. This assures the SAP Gateway, that the request is coming from the same browser or application. Another level of security that you get, along with the X-CSRF-Token.

Only a login (Authorization) is required for a GET operation, because it is Read only. Because the other activities (Create, Update and Delete, a.k.a. "CUD") are performing changes in your backend system, more robust authentication is required. If you search on the community websites for the above, you'll see lots of confusion, and people trying to disable it. Don't disable it, just play by the rules and figure it out. Hopefully this section will save you some time and hardship. Here is an overview of what we will need to do:


External Application
SAP Gateway
Fetch Authorization Data
HTTP GET
SAP Login (Authorization)
Generate X-CSRF-Token
Generate Cookie
HTTP Result
Database Change (CUD Operation)
HTTP POST, PUT, or DELETE
Validate Token and Cookie
Execute Service CUD Operation
HTTP Result

Fetch Authorization Data

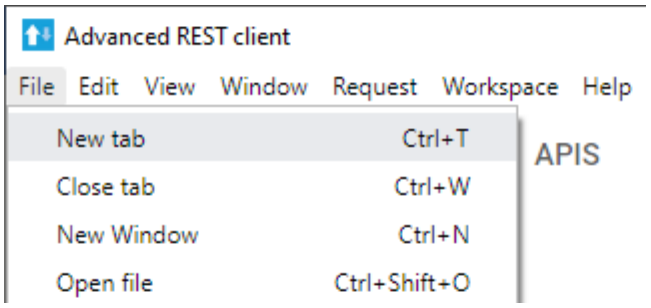
Here is the detailed information, for each process step, above, when using the “Advanced REST Client” software.

First, we need to obtain an encrypted Basic Authorization string, which will contain our SAP Logon ID and Password. This allows us to auto-login to the SAP Gateway programmatically. In addition to the below instructions, I’ve also included instructions on obtaining the Authorization string using IntelliJ IDE in the Reference section at the end of this blog.

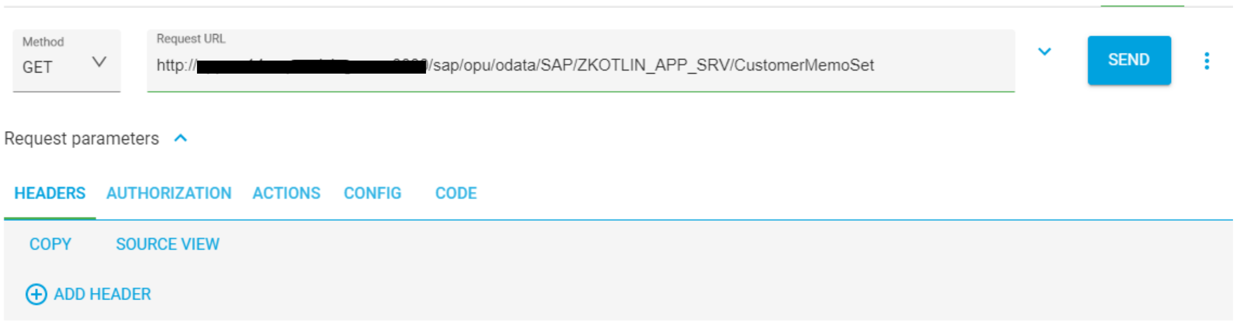
Open the ARC (Advanced REST Client) testing tool. Enter the same URL, which we were testing for our GET earlier:

http://<your server name>/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

In your ARC tool, select the menu path “File > New tab”:

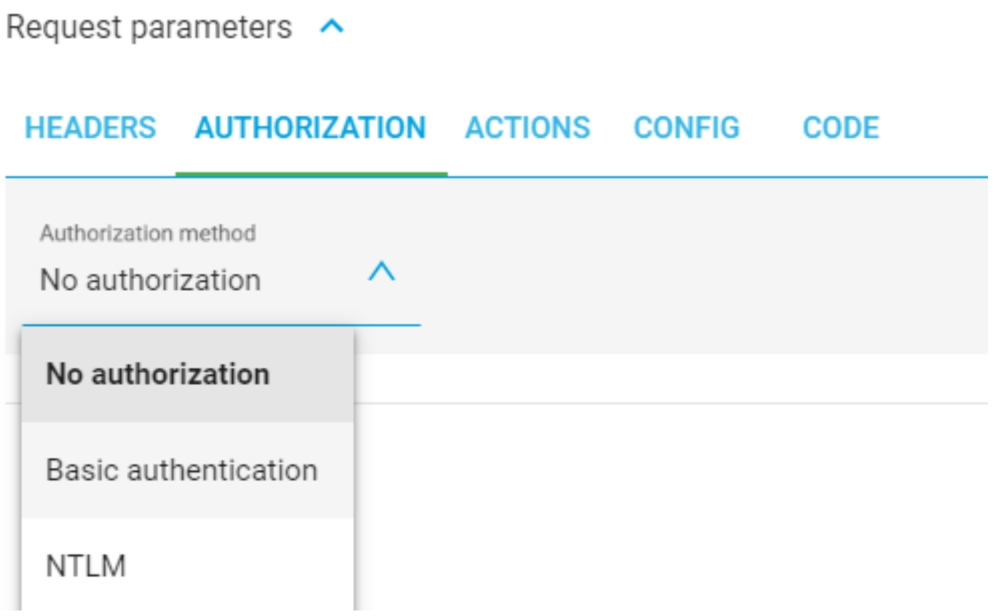


Choose the Method "GET" and enter the url:

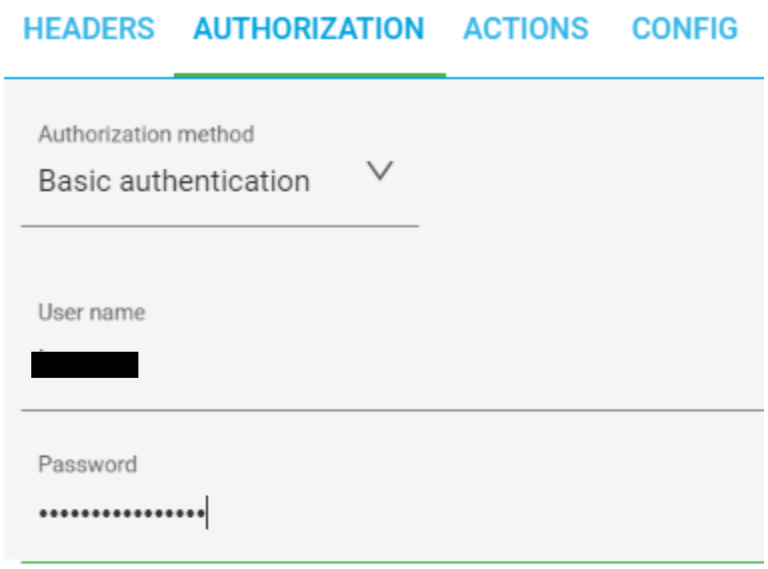

Method: GET

Request URL: http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

Click on the "Authorization" Tab and choose "Basic authentication":


Authorization method: No authorization

Enter your SAP ID and password:

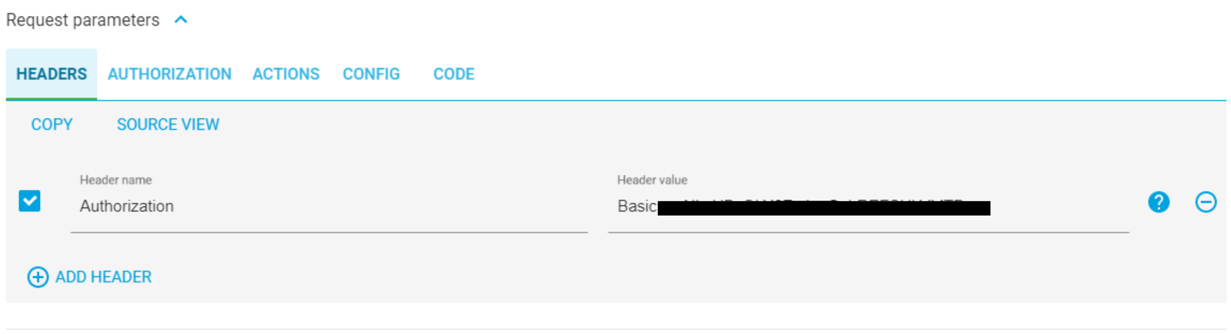


Authorization method
Basic authentication

User name
[redacted]

Password
••••••••••••••••

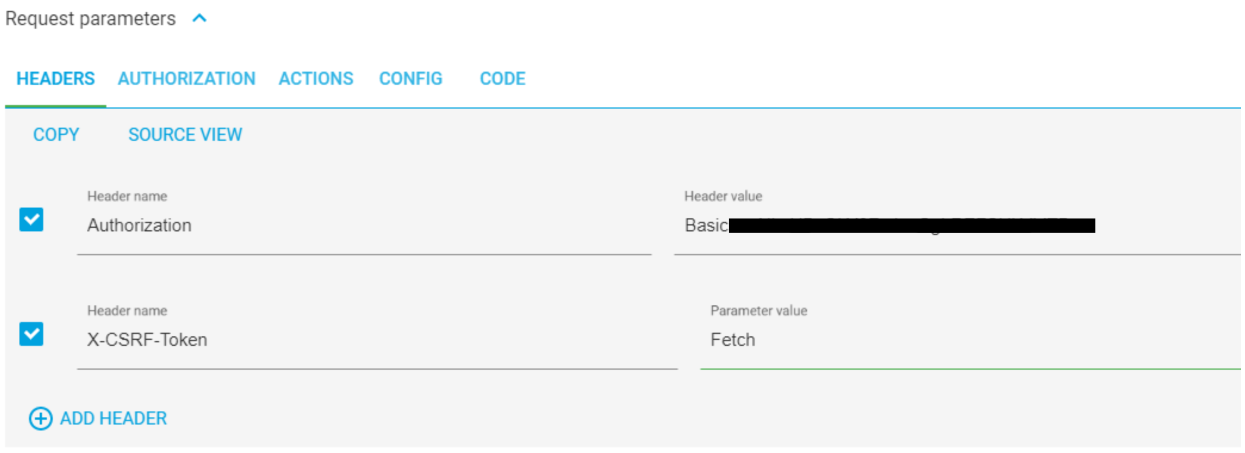
Click on the “HEADERS” tab, and you will see how this encrypted string will be passed in the header of the HTTP request:



Header name	Header value
Authorization	Basic [redacted]

Click “ADD HEADER” to add another header, and enter the following:

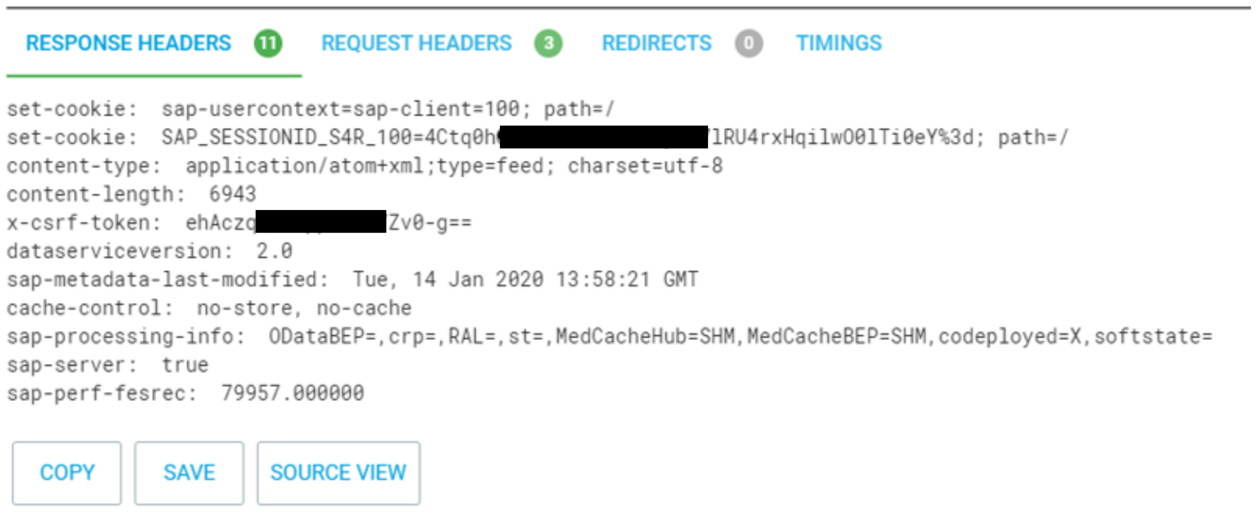
Name: X-CSRF-Token
Value: Fetch



Header name	Header value
Authorization	Basic [redacted]
X-CSRF-Token	Fetch

Now, click on the “SEND” button to perform the HTTP GET. Expand out the “DETAILS” on the right-hand side:

Note the RESPONSE HEADERS will contain the Cookie and Token:



```
set-cookie:  sap-usercontext=sap-client=100; path=/
set-cookie:  SAP_SESSIONID_S4R_100=4Ctq0h[redacted]1RU4rxHqilwO0lTi0eY%3d; path=/
content-type:  application/atom+xml;type=feed; charset=utf-8
content-length:  6943
x-csrf-token:  ehAczq[redacted]Zv0-g==
dataserviceversion:  2.0
sap-metadata-last-modified:  Tue, 14 Jan 2020 13:58:21 GMT
cache-control:  no-store, no-cache
sap-processing-info:  ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,MedCacheBEP=SHM,codeployed=X,softstate=
sap-server:  true
sap-perf-fesrec:  79957.000000
```

Copy the string from the second “set-cookie”:

```
 SAP_SESSIONID_S4R_100=4Ctq0h<blah, blah, blah>lRU4rxHqilwO0lTi0eY%3d; path=/
Copy the string for the “x-csrf-token”:
ehAczq<blah, blah, blah>Zv0-g==
```

These are the 3 bits of information we need, to perform the database update operations (Create, Update or Delete), below:

1. Basic Auth String
2. Cookie String
3. CSRF Token String

Copy the 3 above strings, then proceed below…

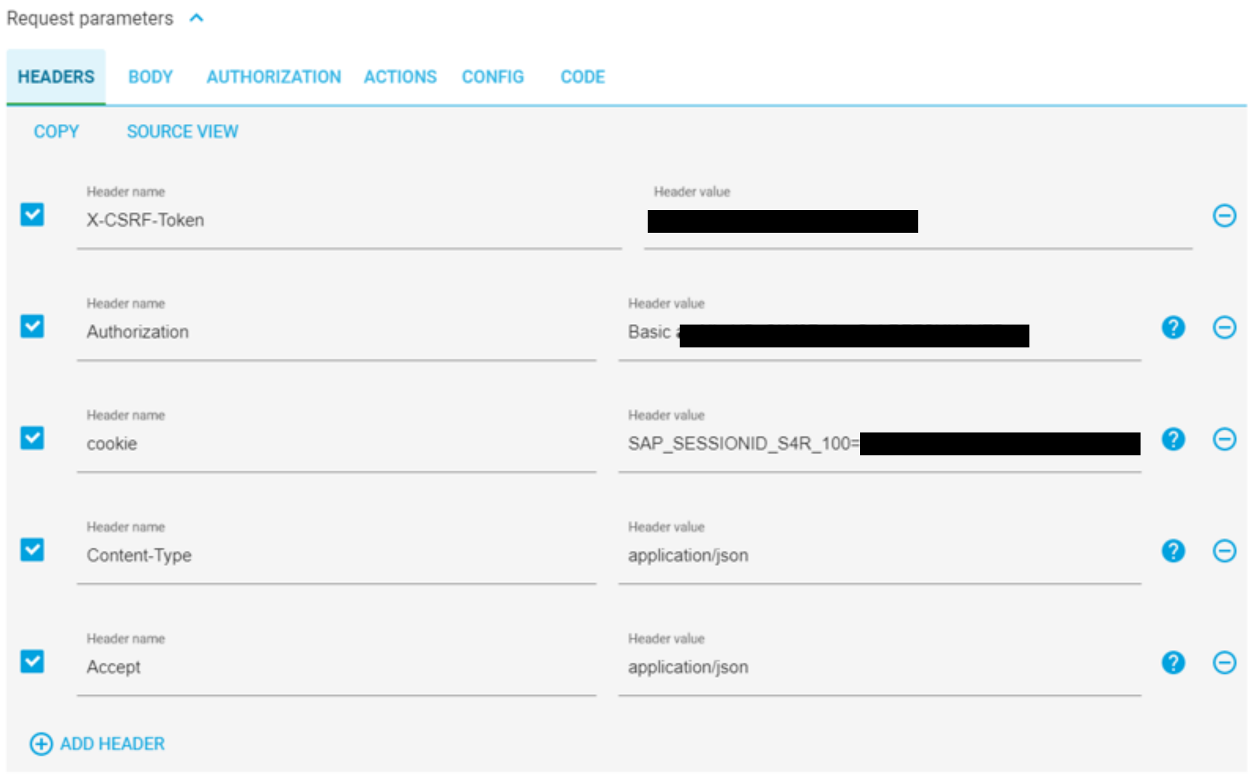
Database Change (CUD Operation)

Using our 3 HTTP header strings we obtained with the GET (Fetch), above, lets perform the same steps we did with the SAP Gateway Client to do a Create (HTTP POST) operation.

NOTE! THE TOKEN AND COOKIE CAN EXPIRE ON THE SERVER, SO IF IT'S BEEN A WHILE SINCE YOU DID THE ABOVE GET, THEN RUN IT AGAIN AND GET A NEW TOKEN AND COOKIE, BEFORE PROCEEDING.

Add a new Tab (Request Editor) in the ARC:

Copy our 3 strings, from the above GET and add them as new Headers. Also, add the below additional "Content-Type" and "Accept" strings. Here is the complete HTTP Header set:

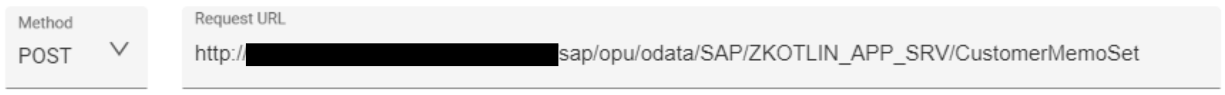

Request parameters

Header name	Header value
X-CSRF-Token	[redacted]
Authorization	Basic [redacted]
cookie	SAP_SESSIONID_S4R_100=[redacted]
Content-Type	application/json
Accept	application/json

The strings above are:

X-CSRF-Token: <your token from the above GET result>
Authorization: <your Basic auth string, used in the above GET section>
cookie: <your cookie from the above GET result>
Content-Type: application/json
Accept: application/json

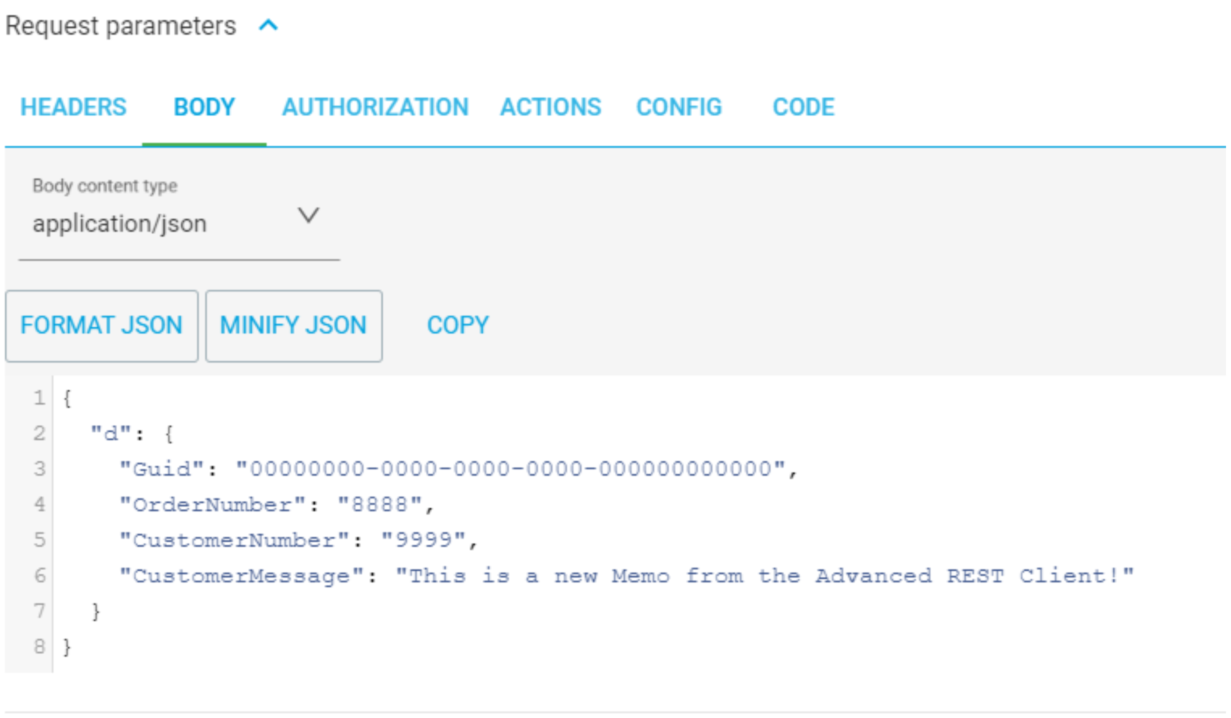
Now, switch the Method to "POST" and enter the following URL, same as the one earlier when testing on the SAP Gateway Client:


Method: POST

Request URL: http:/[redacted]sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet

Now, go to the "Body" tab, and enter the following JSON:

```
{
  "d": {
    "Guid": "00000000-0000-0000-0000-000000000000",
    "OrderNumber": "8888",
    "CustomerNumber": "9999",
    "CustomerMessage": "This is a new Memo from the Advanced REST Client!"
  }
}
```

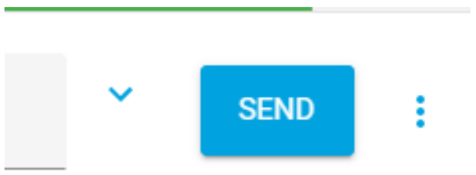


Body content type
application/json

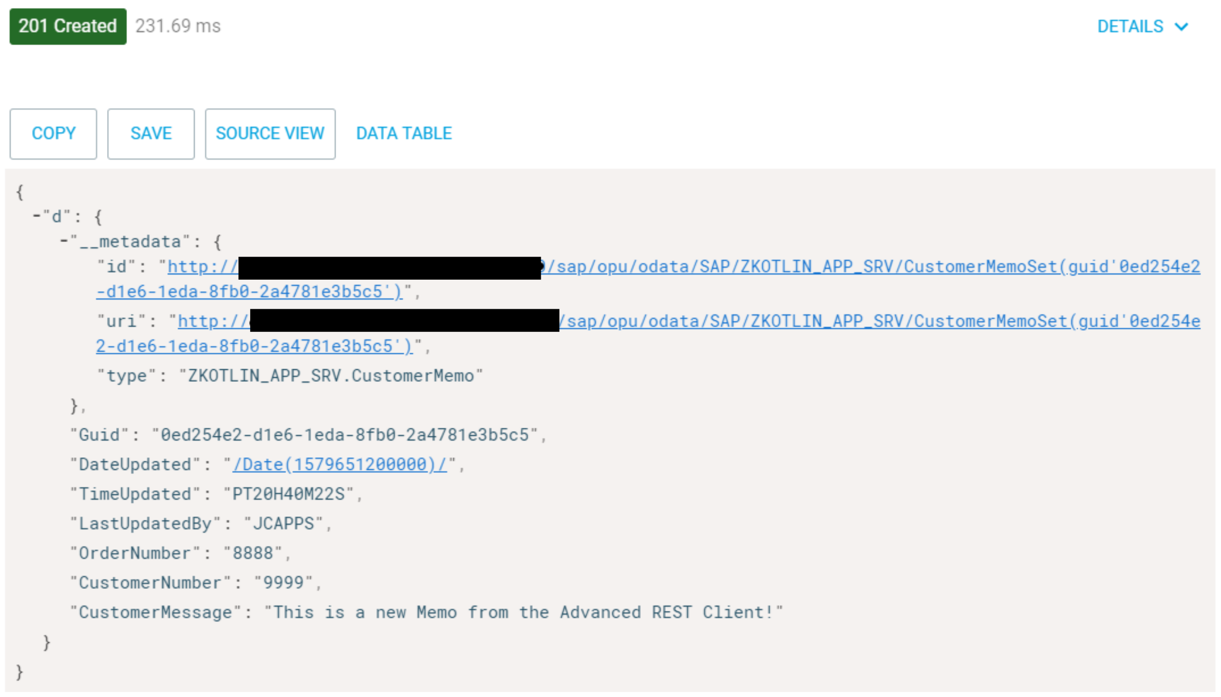
```
{
  "d": {
    "Guid": "00000000-0000-0000-0000-000000000000",
    "OrderNumber": "8888",
    "CustomerNumber": "9999",
    "CustomerMessage": "This is a new Memo from the Advanced REST Client!"
  }
}
```

Since the Guid will be ignored, we could place all zeros in it, but be sure to follow the exact same format, including the dashes (-) and the same length.

Press the “SEND” button to execute the HTTP POST to create our new record:

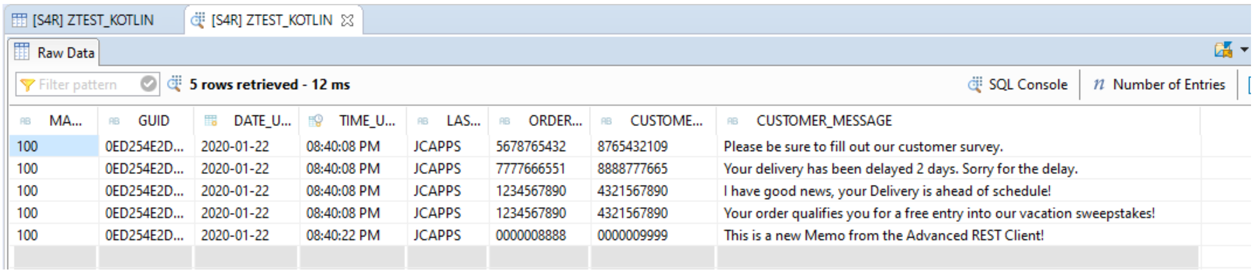


If all goes well, you should see the following success result, including the newly created record:



```
{
  -"d": {
    -"__metadata": {
        "id": "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2
        -d1e6-1eda-8fb0-2a4781e3b5c5')",
        "uri": "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e
        2-d1e6-1eda-8fb0-2a4781e3b5c5')",
        "type": "ZKOTLIN_APP_SRV.CustomerMemo"
     },
     "Guid": "0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5",
     "DateUpdated": "/Date(1579651200000)/",
     "TimeUpdated": "PT20H40M22S",
     "LastUpdatedBy": "JCAPPS",
     "OrderNumber": "8888",
     "CustomerNumber": "9999",
     "CustomerMessage": "This is a new Memo from the Advanced REST Client!"
   }
}
```

To see the above record, you can copy the newly created Guid Key, from the above results, and do another GET from either the SAP Gateway Client or ARC. Here is a complete table listing, from Eclipse, with our new record:

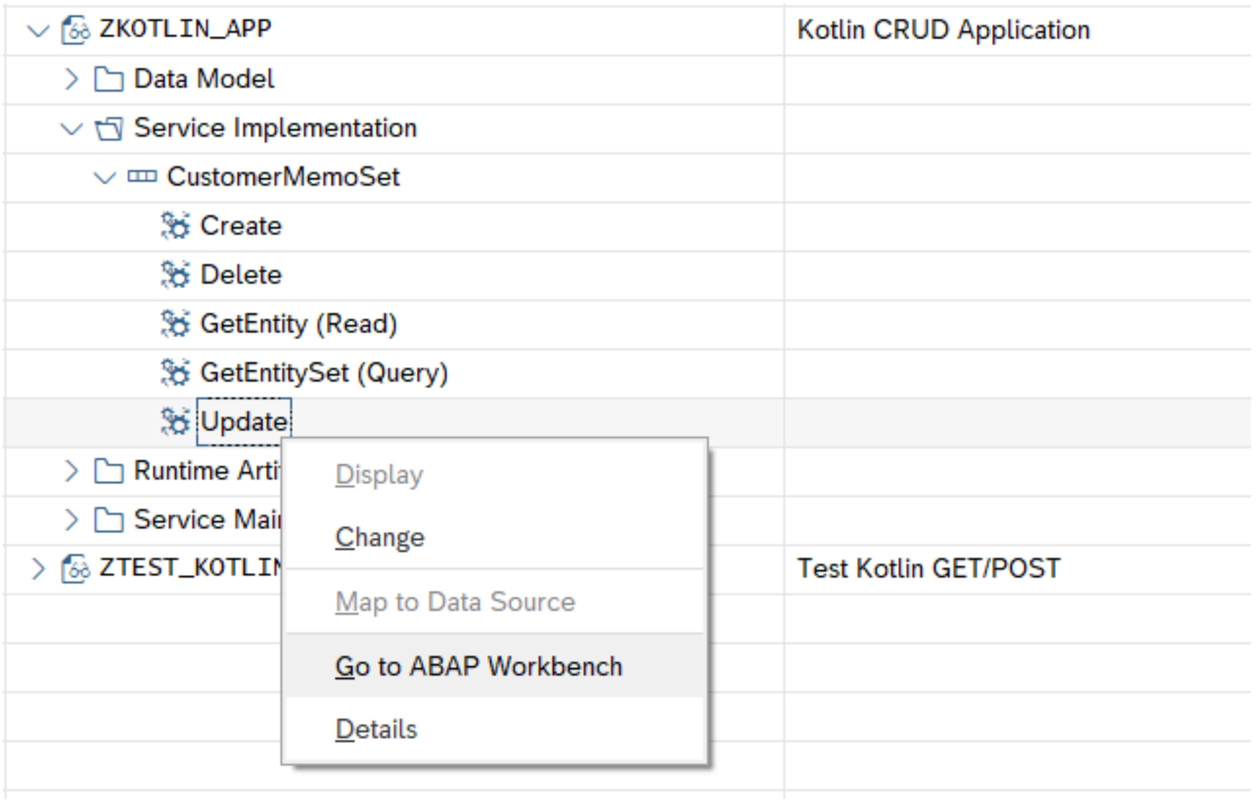


MA...	GUID	DATE_U...	TIME_U...	LAS...	ORDER...	CUSTOME...	CUSTOMER_MESSAGE
100	0ED254E2D...	2020-01-22	08:40:08 PM	JCAPPS	5678765432	8765432109	Please be sure to fill out our customer survey.
100	0ED254E2D...	2020-01-22	08:40:08 PM	JCAPPS	7777666551	8888777665	Your delivery has been delayed 2 days. Sorry for the delay.
100	0ED254E2D...	2020-01-22	08:40:08 PM	JCAPPS	1234567890	4321567890	I have good news, your Delivery is ahead of schedule!
100	0ED254E2D...	2020-01-22	08:40:08 PM	JCAPPS	1234567890	4321567890	Your order qualifies you for a free entry into our vacation sweepstakes!
100	0ED254E2D...	2020-01-22	08:40:22 PM	JCAPPS	0000008888	0000009999	This is a new Memo from the Advanced REST Client!

At this point, we've tested first with the SAP Gateway Client, to prove our GET and POST methods worked correctly. Next, we used a REST tool, called "Advanced REST Client" (ARC) to ensure our methods worked when calling from an external application, with necessary authorization tokens. We can now confidently tell our Kotlin developers that our SAP Gateway service is ready to be called and has been unit tested. Before giving them the green light, let's complete the Update and Delete operations.

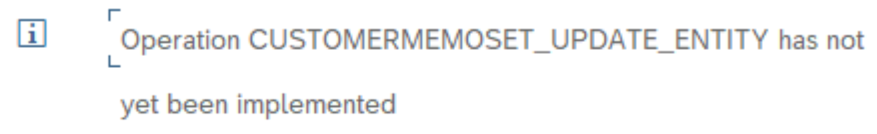
Update Record

Go back to the SAP Gateway Service Builder (Transaction SEGW), and go to the following node and right-click on the “Update” method, then “Go to ABAP Workbench”:



ZKOTLIN_APP	Kotlin CRUD Application
Data Model	
Service Implementation	
CustomerMemoSet	
Create	
Delete	
GetEntity (Read)	
GetEntitySet (Query)	
Update	
Runtime Arti	
Service Mai	
ZTEST_KOTLIN	Test Kotlin GET/POST

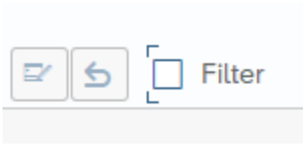
You’ll get the warning that it’s not yet implemented:



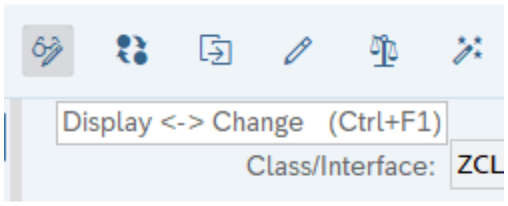
Operation CUSTOMERMEMOSET_UPDATE_ENTITY has not yet been implemented

Enter to continue.

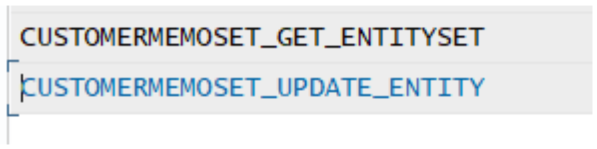
Uncheck the “Filter” checkbox, if it’s on:



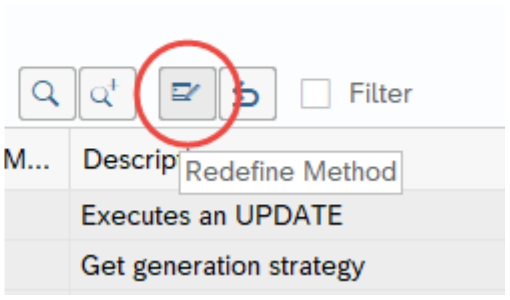
Go into Change Mode for the Class:



Single click into the row for the method CUSTOMERMEMOSET_UPDATE_ENTITY:


CUSTOMERMEMOSET_GET_ENTITYSET
CUSTOMERMEMOSET_UPDATE_ENTITY

Click on the “Redefine Method” button:



Executes an UPDATE
Get generation strategy

Overwrite the commented code in the method, with the following code:

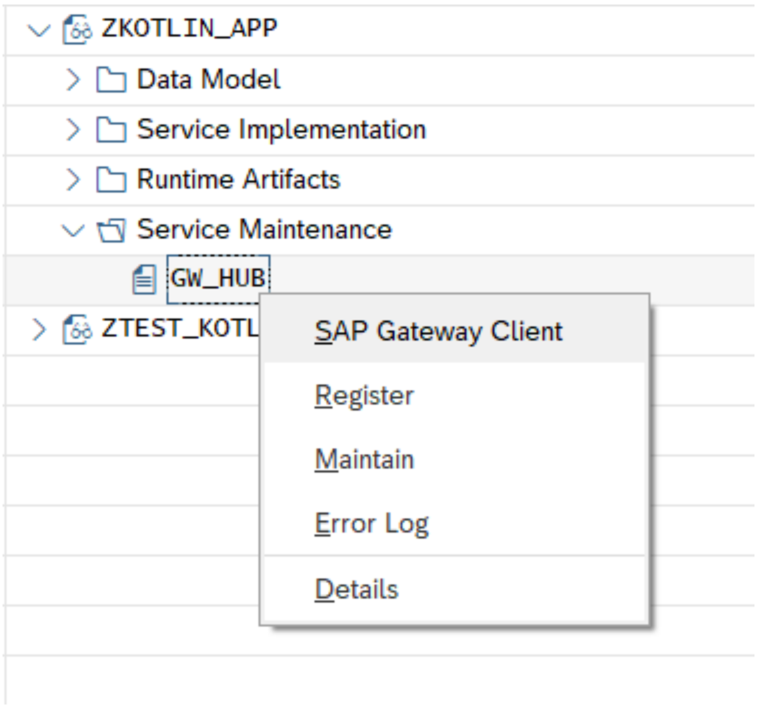
```
method CUSTOMERMEMOSET_UPDATE_ENTITY.
  DATA: lw_record TYPE zcl_zkotlin_app_mpc=>ts_customermemo.

  io_data_provider->read_entry_data( IMPORTING es_data = lw_record ).

  "The key is "guid"...
  READ TABLE it_key_tab INTO DATA(lw_key) INDEX 1.

  "Make sure the value matches the one in the OData payload...
  IF lw_key-value = lw_record-guid.
    "Update the record in the database...
    CLEAR: lw_record-date_updated, lw_record-time_updated, lw_record-last_updated_by.
    lw_record-date_updated    = sy-datum.
    lw_record-time_updated    = sy-uzeit.
    lw_record-last_updated_by = sy-uname.
    UPDATE ztest_kotlin FROM lw_record.
    COMMIT WORK AND WAIT.
    IF sy-subrc = 0.
      "Entity was found and updated...
      er_entity = lw_record.
    ELSE.
      "Entity not found
      RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
        EXPORTING
          textid      = /iwbep/cx_mgw_busi_exception=>resource_not_found
          entity_type = |{ iv_entity_name } ('{ lw_key-value }')|.
    ENDIF.
  ENDIF.
endmethod.
```

Save and Activate your code.

For testing, we will repeat the same steps as shown in the above “Create Record” section, except we will do an HTTP PUT, instead of an HTTP POST. Plus, we’ll have to pass the complete record in the HTTP payload, along with the Guid key for the record we want to update. The testing steps in this section are more summarized.

For a very detailed explanation of the steps to test an HTTP operation with a database update, see the "Create Record" section, above.

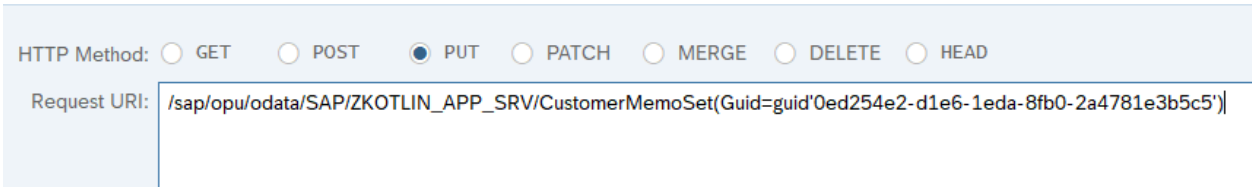
Go to the SAP Gateway Client:



Retrieve the Guid key, for any existing record you want to update.

Enter the request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(Guid=guid'0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5')

Select the PUT method:

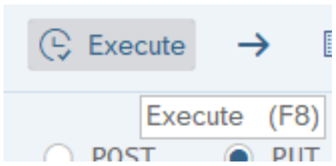

HTTP Method: GET POST [✓ PUT] PATCH MERGE DELETE HEAD

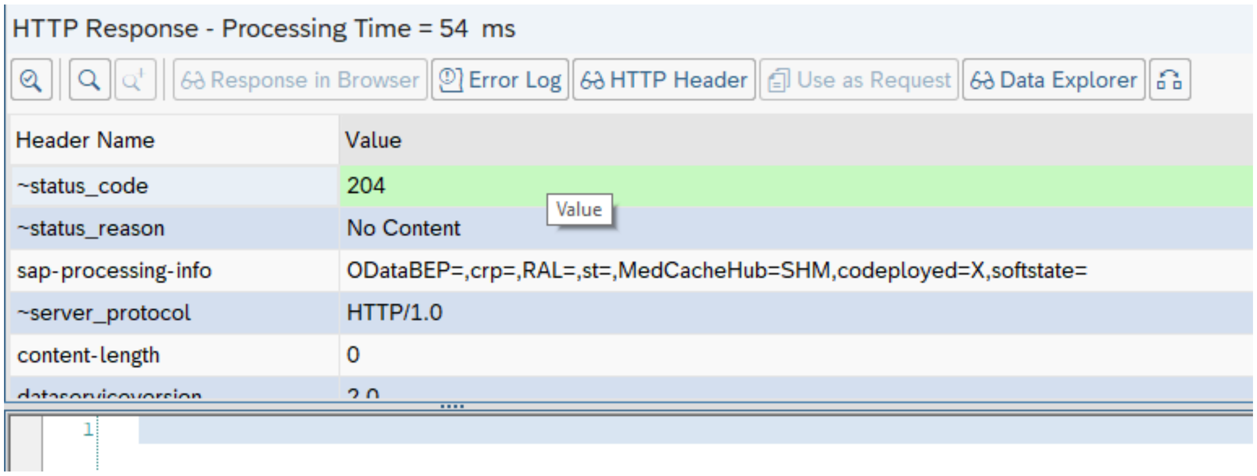
Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(Guid=guid'0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5')

Enter your JSON in the HTTP Request payload section:

```
{
  "d" : {
    "Guid" : "0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5",
    "DateUpdated" : "\/Date(1578528000000)\/",
    "TimeUpdated" : "PT13H10M38S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "8888",
    "CustomerNumber" : "9999",
    "CustomerMessage" : "This is my updated Memo!"
  }
}
```

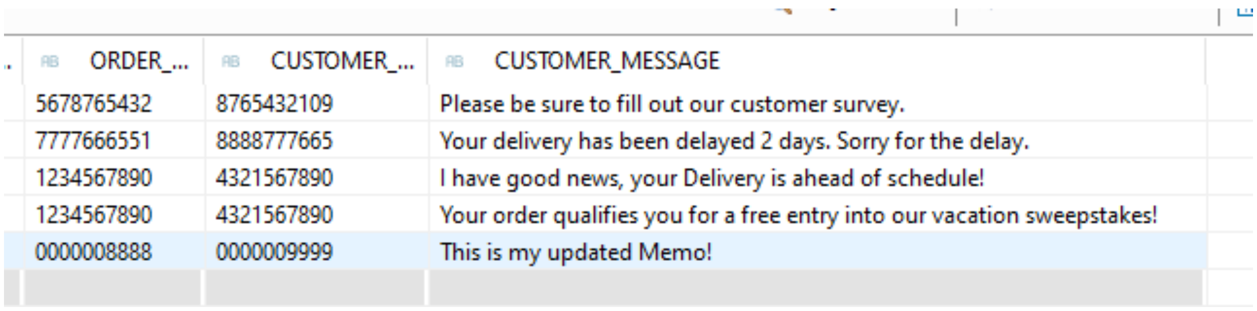
Execute, and you will get the following results:




HTTP Response - Processing Time = 54 ms

Header Name	Value
~status_code	204
~status_reason	No Content
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=
~server_protocol	HTTP/1.0
content-length	0
dataserviceversion	2.0

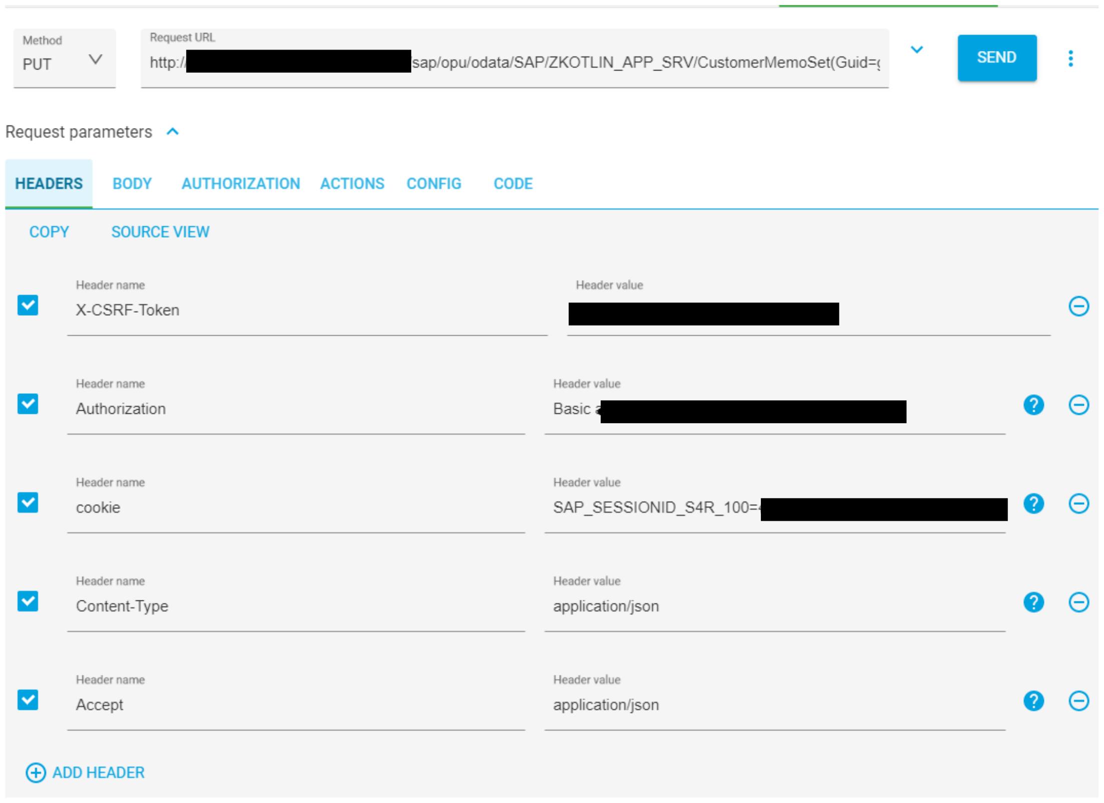
Confirm that the record update was successful:



ORDER_...	CUSTOMER_...	CUSTOMER_MESSAGE
5678765432	8765432109	Please be sure to fill out our customer survey.
7777666551	8888777665	Your delivery has been delayed 2 days. Sorry for the delay.
1234567890	4321567890	I have good news, your Delivery is ahead of schedule!
1234567890	4321567890	Your order qualifies you for a free entry into our vacation sweepstakes!
0000008888	0000009999	This is my updated Memo!

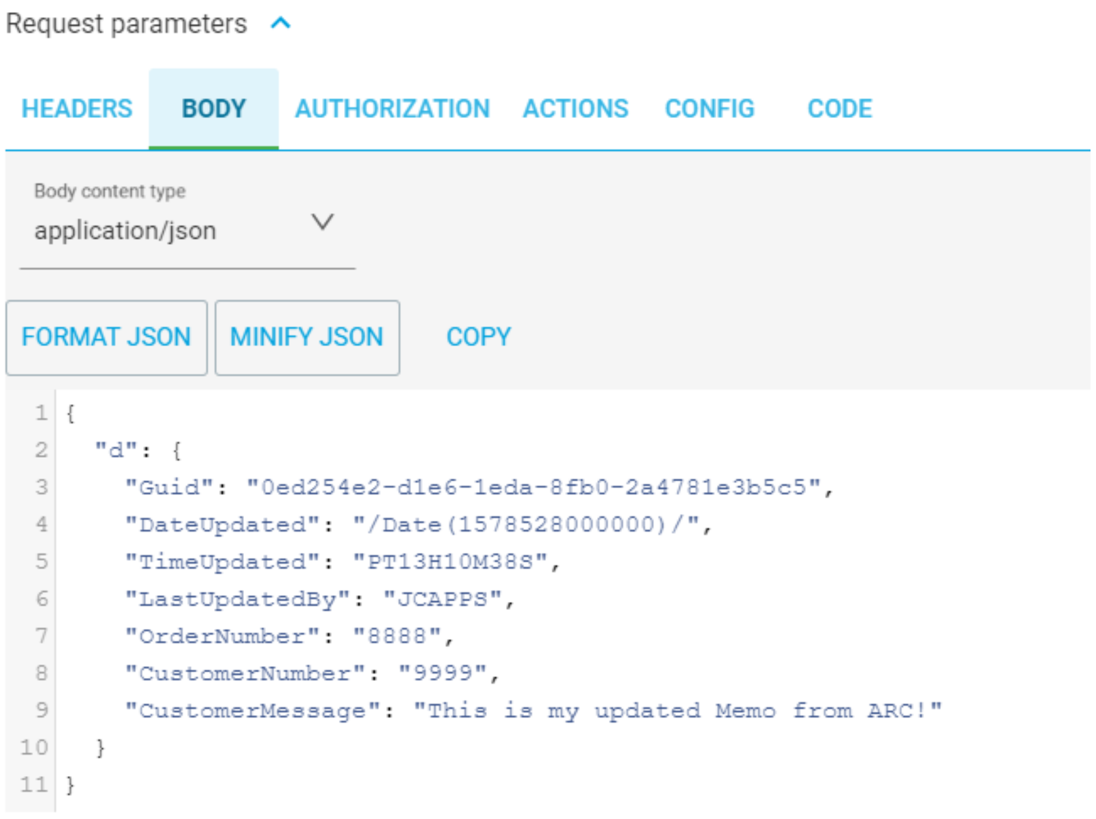
To perform this test with the Advanced REST Client (ARC), perform the same steps in the “Database Change (CUD Operation)” section, above, but include the above payload, and your Guid Key in the url.

As with the Create Operation, perform a GET first, to fetch your authorization token data. Then, do an HTTP PUT Method:



Request URL: http:/[redacted]sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(Guid=(

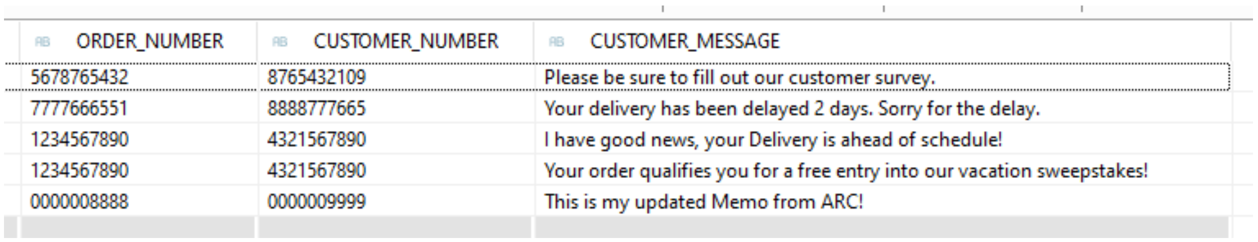
Header name	Header value
X-CSRF-Token	[redacted]
Authorization	Basic [redacted]
cookie	SAP_SESSIONID_S4R_100=[redacted]
Content-Type	application/json
Accept	application/json



Body content type: application/json

```
{
  "d": {
    "Guid": "0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5",
    "DateUpdated": "/Date(1578528000000)/",
    "TimeUpdated": "PT13H10M38S",
    "LastUpdatedBy": "JCAPPS",
    "OrderNumber": "8888",
    "CustomerNumber": "9999",
    "CustomerMessage": "This is my updated Memo from ARC!"
  }
}
```

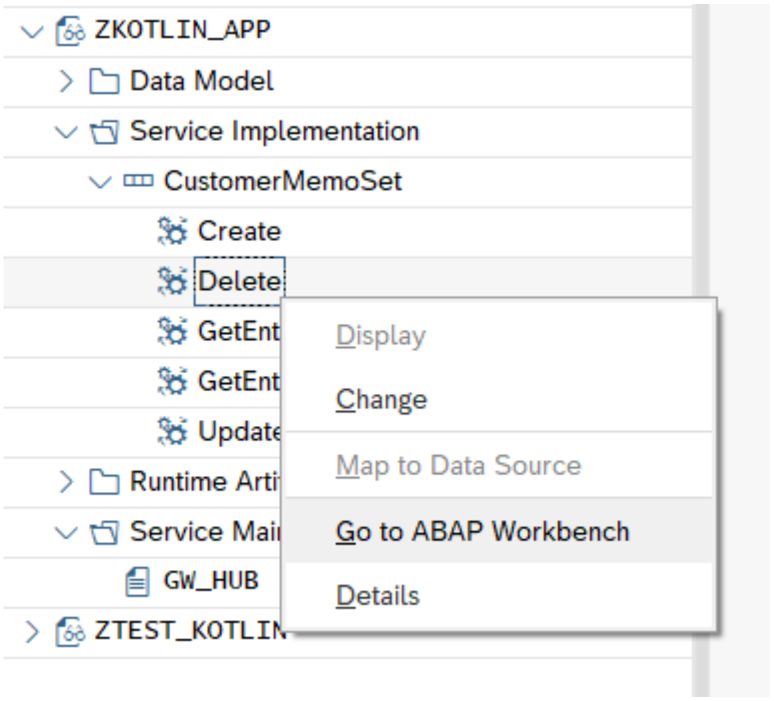
Click the "SEND" button to execute the above record update.



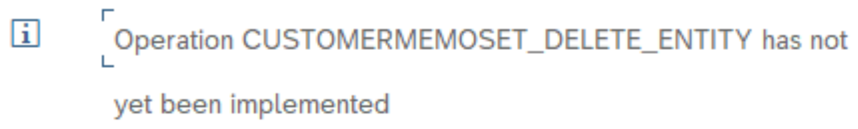
ORDER_NUMBER	CUSTOMER_NUMBER	CUSTOMER_MESSAGE
5678765432	8765432109	Please be sure to fill out our customer survey.
7777666551	8888777665	Your delivery has been delayed 2 days. Sorry for the delay.
1234567890	4321567890	I have good news, your Delivery is ahead of schedule!
1234567890	4321567890	Your order qualifies you for a free entry into our vacation sweepstakes!
0000008888	0000009999	This is my updated Memo from ARC!

DELETE RECORD

Go back to the SAP Gateway Service Builder (Transaction SEGW), and go to the following node and right-click on the "Delete" method, then "Go to ABAP Workbench":



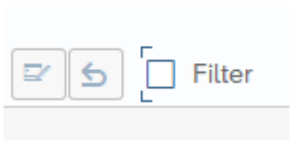
You'll get the warning that it's not yet implemented:



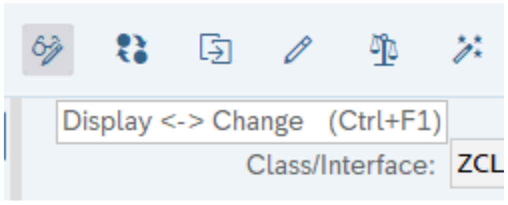
Operation CUSTOMERMEMOSET_DELETE_ENTITY has not yet been implemented

Enter to continue.

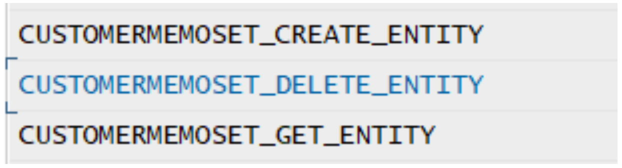
Uncheck the "Filter" checkbox, if it's on:



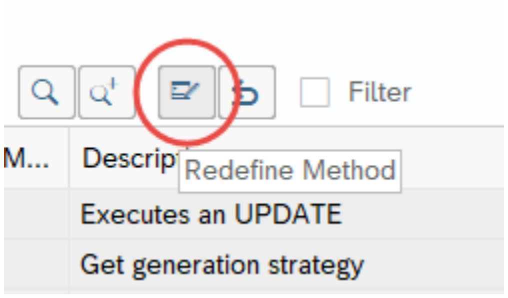
Go into Change Mode for the Class:


Display <-> Change (Ctrl+F1)
Class/Interface: ZCL

Single click into the row for the method CUSTOMERMEMOSET_DELETE_ENTITY:


CUSTOMERMEMOSET_CREATE_ENTITY
CUSTOMERMEMOSET_DELETE_ENTITY
CUSTOMERMEMOSET_GET_ENTITY

Click on the “Redefine Method” button:



Executes an UPDATE
Get generation strategy

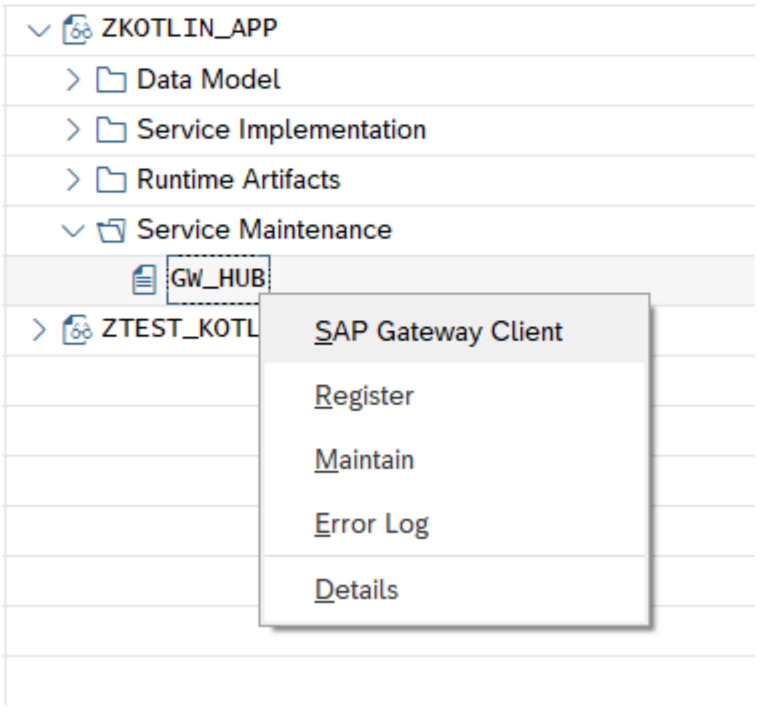
Overwrite the commented code in the method, with the following code:

```
METHOD customermemoset_delete_entity.
  DATA(lt_keys) = io_tech_request_context->get_keys( ).

  READ TABLE lt_keys WITH KEY name = 'GUID' INTO DATA(ls_key).
  DATA(lv_guid) = ls_key-value.

  "Ensure the GUID is valid...
  SELECT SINGLE mandt INTO @DATA(l_mandt) FROM ztest_kotlin
    WHERE guid = @lv_guid.
  IF sy-subrc <> 0.
    "Record not found...
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
      EXPORTING
        textid      = /iwbep/cx_mgw_busi_exception=>resource_not_found
        entity_type = |{ iv_entity_name } (Guid='{ lv_guid }')|.
  ENDIF.
  "Delete the record from the database...
  DELETE FROM ztest_kotlin WHERE guid = lv_guid.
  COMMIT WORK AND WAIT.
  IF sy-subrc <> 0.
    "Delete Failed...
    RAISE EXCEPTION TYPE /iwbep/cx_mgw_busi_exception
      EXPORTING
        textid      = /iwbep/cx_mgw_busi_exception=>resource_not_found
        entity_type = |{ iv_entity_name } (Guid='{ lv_guid }')|.
  ENDIF.
ENDMETHOD.
```

Save and Activate your code.

For testing, we will repeat the same steps as shown in the above “Create Record” section, except we will do an HTTP DELETE, instead of an HTTP POST. Because we are simply deleting the record, we only need to provide the Guid key in the URL and no payload is needed. The testing steps in this section are more summarized. For a very detailed explanation of the steps to test an HTTP operation with a database update, see the “Create Record” section, above.

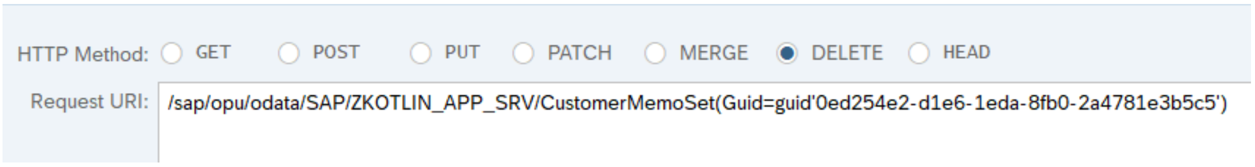
Go to the SAP Gateway Client:



Retrieve the Guid key, for any existing record you want to delete.

Enter the request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(Guid=guid'0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5')

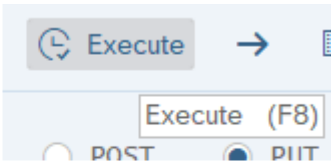
Select the DELETE method:

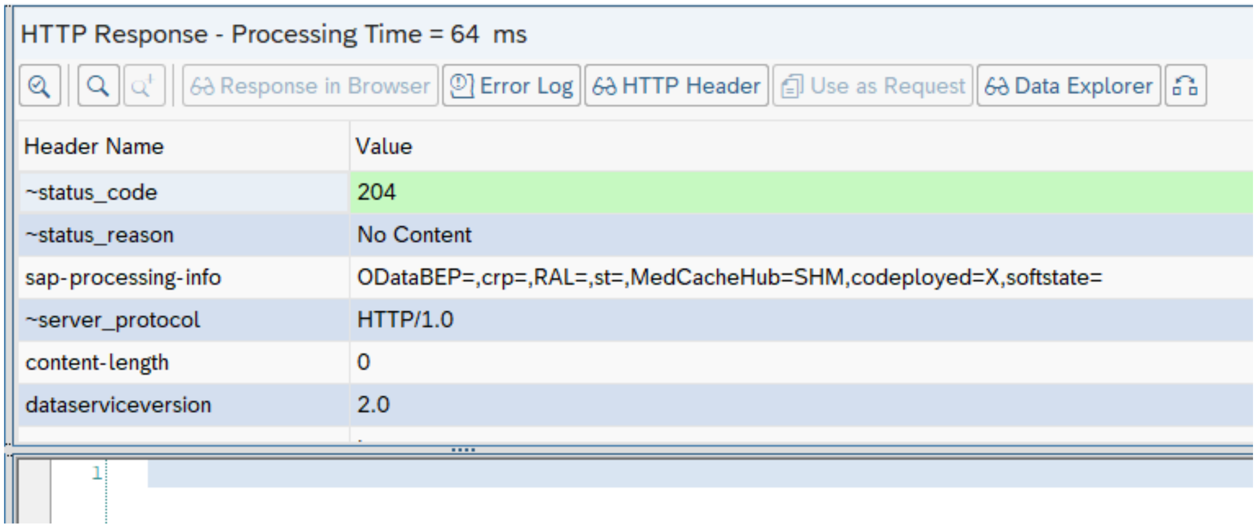


HTTP Method: GET POST PUT PATCH MERGE [✓ DELETE] HEAD

Request URI: /sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(Guid=guid'0ed254e2-d1e6-1eda-8fb0-2a4781e3b5c5')

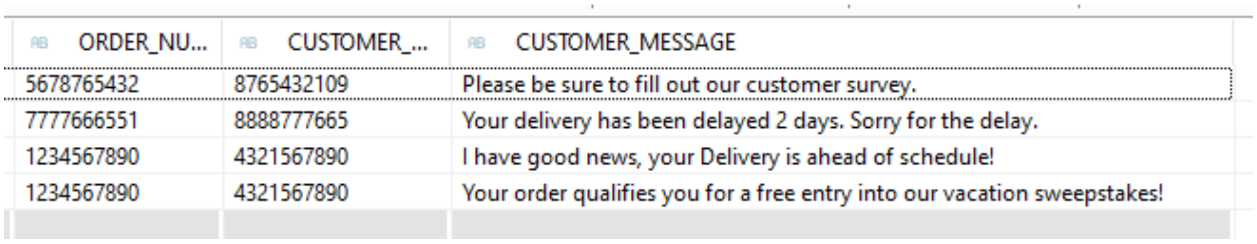
Execute, and you will get the following results:




HTTP Response - Processing Time = 64 ms

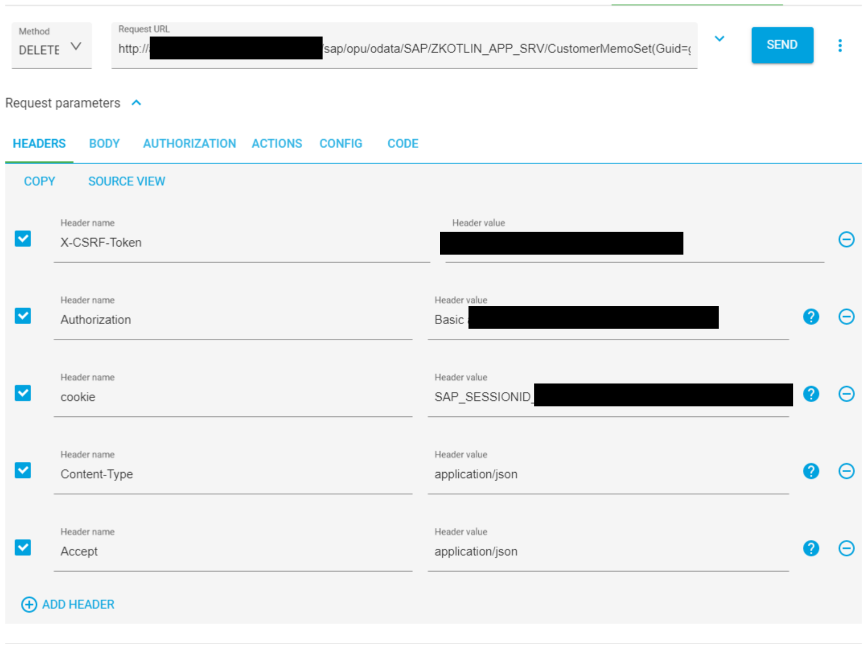
Header Name	Value
~status_code	204
~status_reason	No Content
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=
~server_protocol	HTTP/1.0
content-length	0
dataserviceversion	2.0

Confirm that the record was successfully deleted:



ORDER_NU...	CUSTOMER_...	CUSTOMER_MESSAGE
5678765432	8765432109	Please be sure to fill out our customer survey.
7777666551	8888777665	Your delivery has been delayed 2 days. Sorry for the delay.
1234567890	4321567890	I have good news, your Delivery is ahead of schedule!
1234567890	4321567890	Your order qualifies you for a free entry into our vacation sweepstakes!

To perform this test with the Advanced REST Client (ARC), perform the same steps in the "Database Change (CUD Operation)" section, above, with no payload, and your Guid Key in the url. As with the Create Operation, perform a GET first, to fetch your authorization token data. Then, do an HTTP DELETE Method:



http://[redacted]sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(Guid=(

Header name	Header value
X-CSRF-Token	[redacted]
Authorization	Basic [redacted]
cookie	SAP_SESSIONID_[redacted]
Content-Type	application/json
Accept	application/json

Click the "SEND" button to execute the deletion of the record.

Create a Kotlin Application

At this point, our SAP Gateway Service is fully developed and tested to support all of our CRUD operations for our custom SAP Table. Now, we can write an application using any programming language of our choice, to call the RESTful web service. Of course, Kotlin is the best, so let's begin…

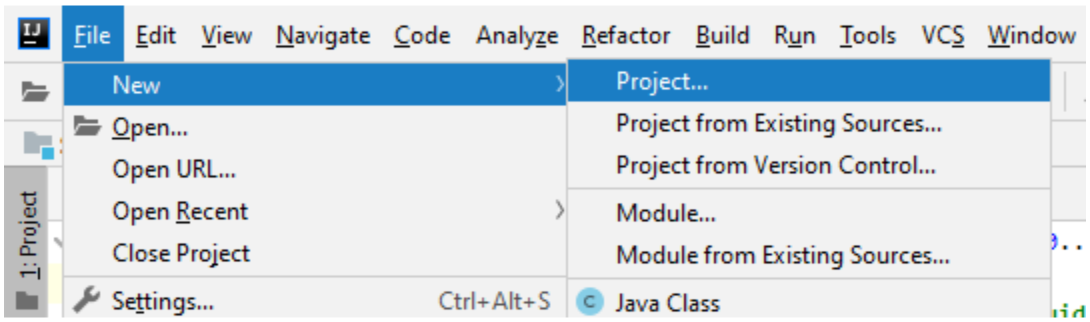
PROJECT CREATION

Kotlin was invented by the folks at JetBrains, who gained fame with their excellent development tools. So, naturally, the best IDE for Kotlin is theirs – IntelliJ IDEA. You may download and install their free community edition from their website:

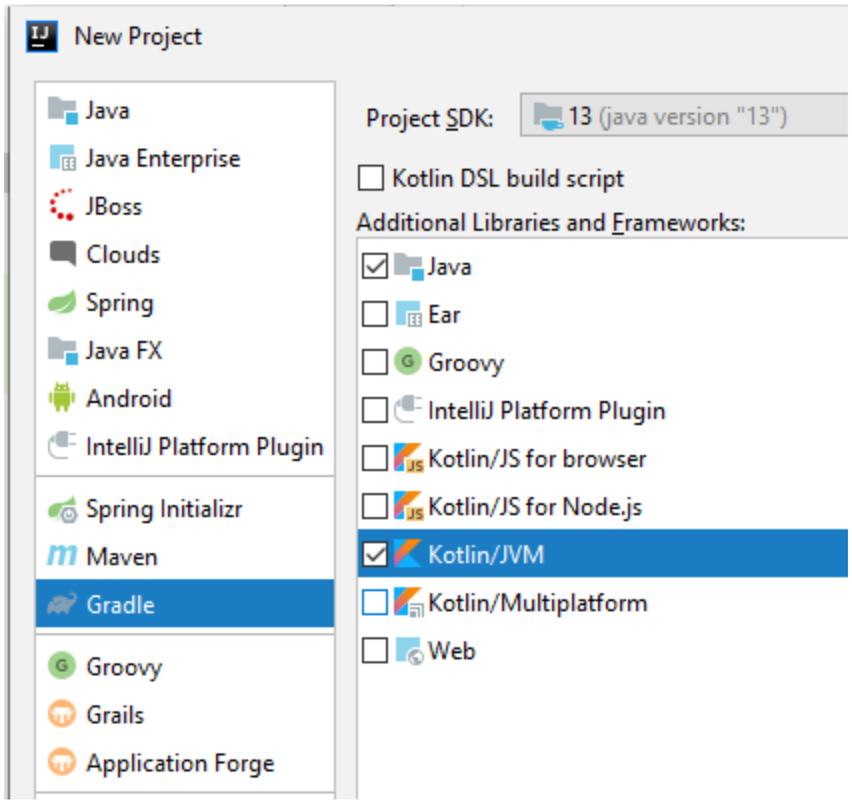
https://www.jetbrains.com/idea/

GRADLE

Gradle is the most compatible with Kotlin, so let's use that for our build tool. Create a new project, as follows:



Select Gradle, Kotlin and Java:



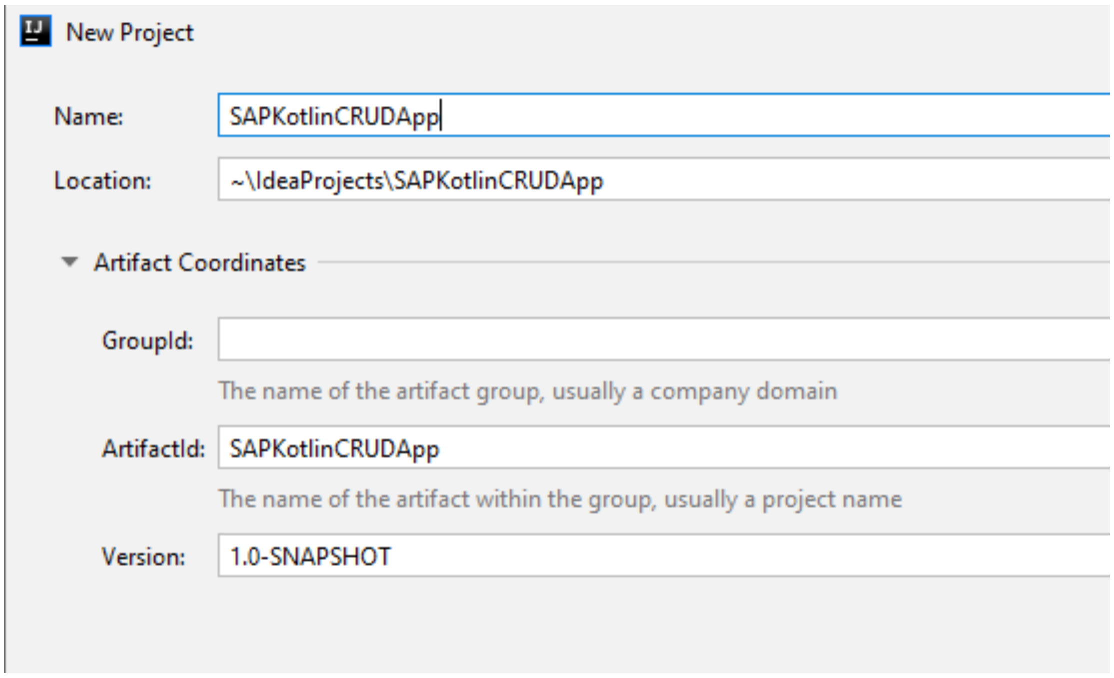
Project SDK: 13 (java version "13")

☐ Kotlin DSL build script

Additional Libraries and Frameworks:

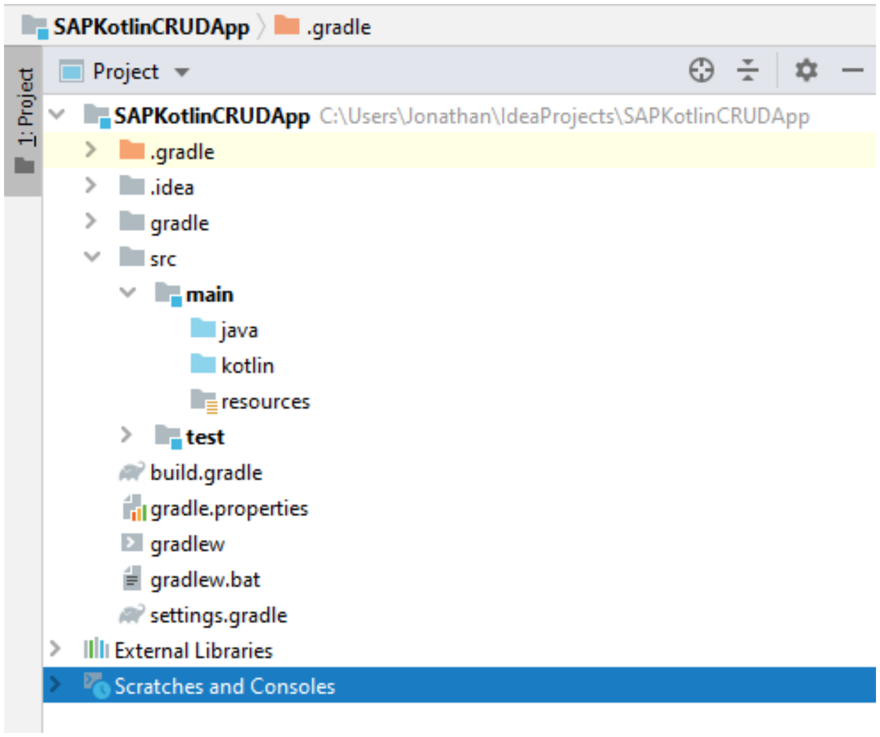
- ✓ Java
- ☐ Ear
- ☐ Groovy
- ☐ IntelliJ Platform Plugin
- ☐ Kotlin/JS for browser
- ☐ Kotlin/JS for Node.js
- ✓ Kotlin/JVM
- ☐ Kotlin/Multiplatform
- ☐ Web

Click Next. Leave GroupId blank, as that is only for a Maven repository. ArtifactId = Project name. Leave Version default:



Name:	SAPKotlinCRUDApp
Location:	~\IdeaProjects\SAPKotlinCRUDApp
GroupId:	
	The name of the artifact group, usually a company domain
ArtifactId:	SAPKotlinCRUDApp
	The name of the artifact within the group, usually a project name
Version:	1.0-SNAPSHOT

Click Finish. This will build the following structure:



Double-click to open the “build.gradle” file, and add the following 2 lines to the “dependencies” section:

```
compile "com.github.kittinunf.fuel:fuel:2.2.1"
compile "com.fasterxml.jackson.module:jackson-module-kotlin:2.9.+"
```

Add the following to the “repositories” section:

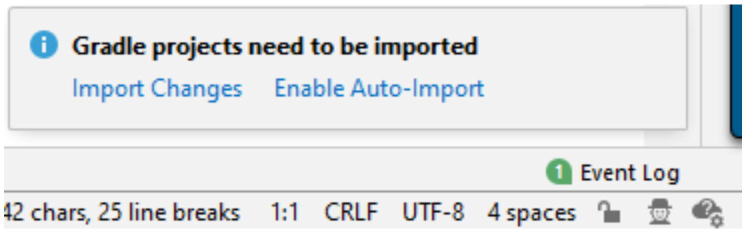
```
jcenter()
```

The complete “build.gradle” file should look like the following:

```
plugins {
    id 'java'
    id 'org.jetbrains.kotlin.jvm' version '1.3.61'
}

version '1.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
    jcenter()
}

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
    testCompile group: 'junit', name: 'junit', version: '4.12'
    compile "com.github.kittinunf.fuel:fuel:2.2.1"
    compile "com.fasterxml.jackson.module:jackson-module-kotlin:2.9.+"
}

compileKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
compileTestKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
```

Your versions could vary, and the code should still work, since it is backward compatible.

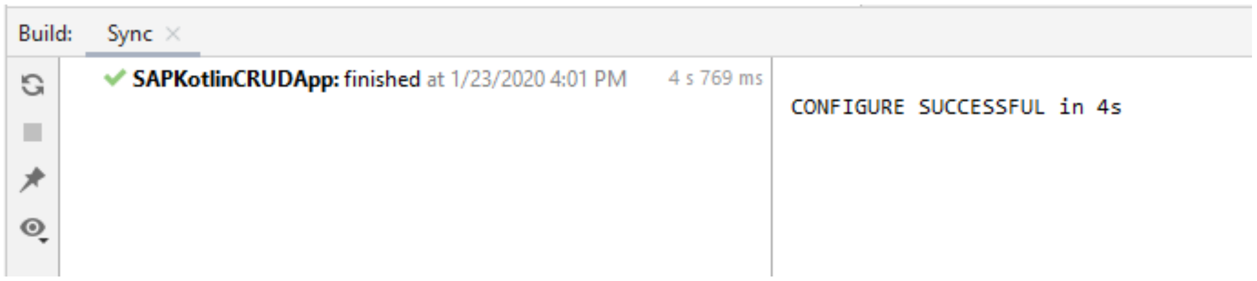
After you add these dependencies, you may see the message in the lower-right:



Gradle projects need to be imported

Import Changes   Enable Auto-Import

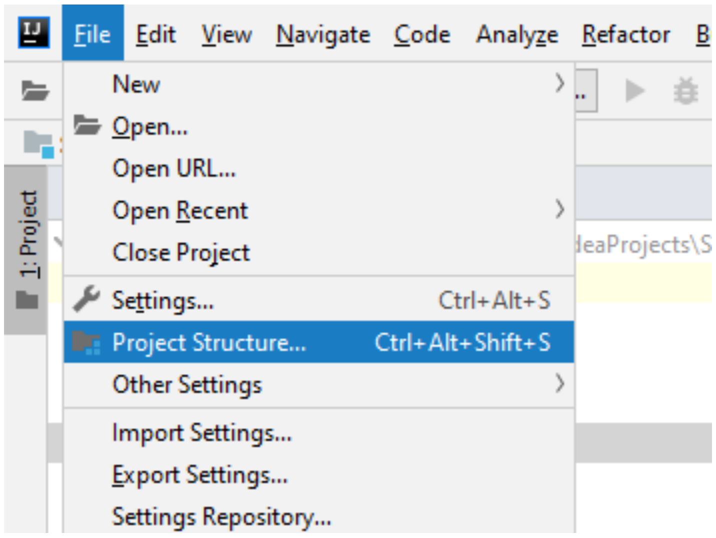
Go ahead and click “Import Changes”. You should get a successful build message:



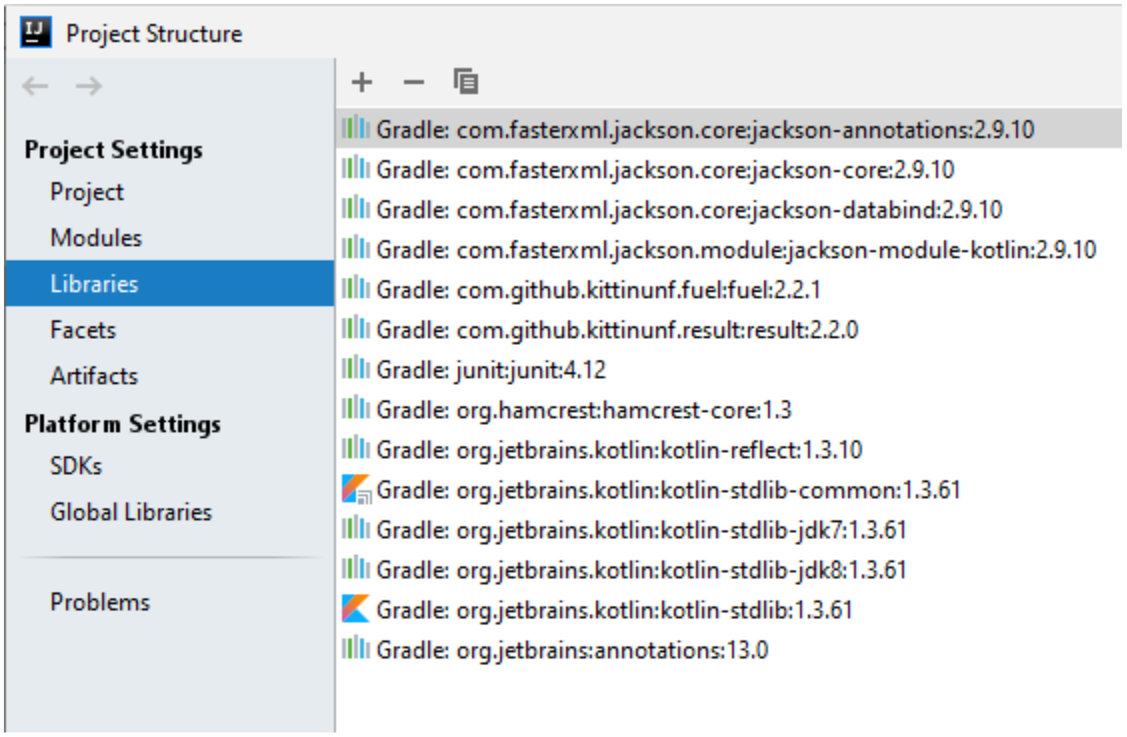
SAPKotlinCRUDApp: finished at 1/23/2020 4:01 PM   4 s 769 ms

```
CONFIGURE SUCCESSFUL in 4s
```

This build.gradle file will pull in all of the libraries for you. Go to the menu path “File > Project Structure”, to see them:



Next, go to the "Libraries" section:



```
Gradle: com.fasterxml.jackson.core:jackson-annotations:2.9.10
Gradle: com.fasterxml.jackson.core:jackson-core:2.9.10
Gradle: com.fasterxml.jackson.core:jackson-databind:2.9.10
Gradle: com.fasterxml.jackson.module:jackson-module-kotlin:2.9.10
Gradle: com.github.kittinunf.fuel:fuel:2.2.1
Gradle: com.github.kittinunf.result:result:2.2.0
Gradle: junit:junit:4.12
Gradle: org.hamcrest:hamcrest-core:1.3
Gradle: org.jetbrains.kotlin:kotlin-reflect:1.3.10
Gradle: org.jetbrains.kotlin:kotlin-stdlib-common:1.3.61
Gradle: org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.3.61
Gradle: org.jetbrains.kotlin:kotlin-stdlib-jdk8:1.3.61
Gradle: org.jetbrains.kotlin:kotlin-stdlib:1.3.61
Gradle: org.jetbrains:annotations:13.0
```

JACKSON AND FUEL

We will be using the Jackson library for all of our JSON parsing needs. If you recall, we can set the SAP Gateway Service to respond and receive data in JSON format. The library may be found here:

https://github.com/FasterXML/jackson

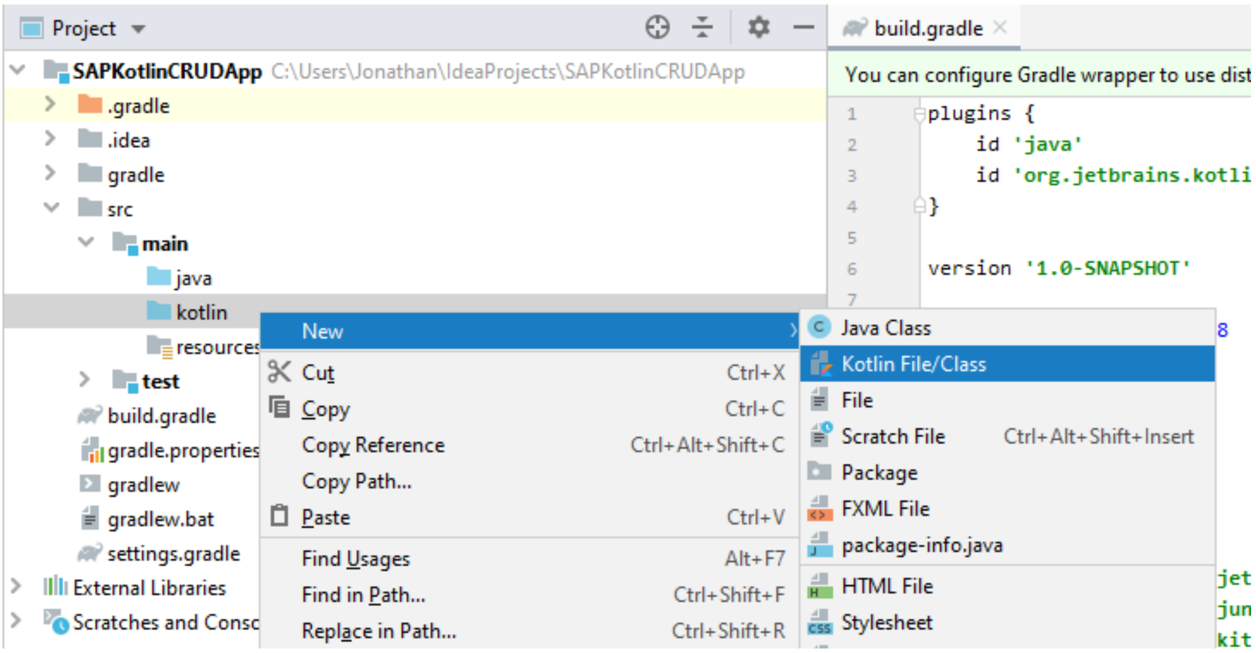
To more easily manage our HTTP networking requests, we'll use the Fuel library. This library may be found here:

https://github.com/kittinunf/fuel

READ RECORDS

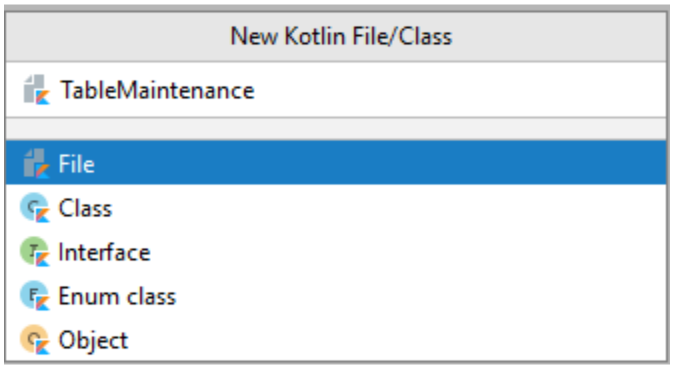
At this point, we will walk through creation of the program step-by-step. If you want to skip the details, then just jump to the end and see the entire source code file.

First, lets write the code to read the data from the SAP Gateway, an HTTP GET. Create a new Kotlin class, by right-clicking on the below “Kotlin” node, then “New > Kotlin File/Class”:



```
plugins {
    id 'java'
    id 'org.jetbrains.kotli
}

version '1.0-SNAPSHOT'
```

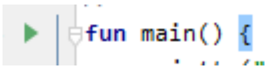
Create a file named “TableMaintenance”, which will serve as our only source code file, and the entry point of our program:



Hit Enter to create the new Kotlin file. Just like Java, the main entry point for our Kotlin program will be called “main”. Create the following method:

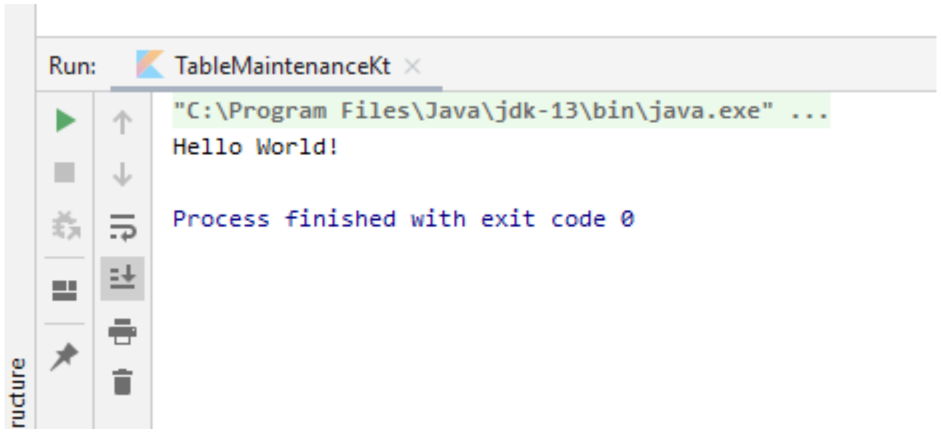
```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method main: Entry point for the program...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
fun main() {
    println("Hello World!")
}
```

Once created, you will see a “Play” icon next to our new method. Click it, to run the program:



```
fun main() {
```

The program will compile and run, and you will get the following results:

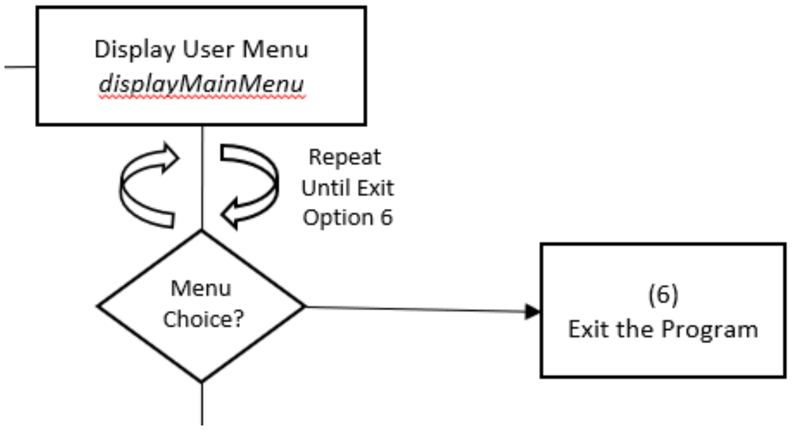


```
"C:\Program Files\Java\jdk-13\bin\java.exe" ...
Hello World!

Process finished with exit code 0
```

Now that we know everything is working, let's create our user menu for our project. Create the following new method, which will display our menu:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method displayMainMenu: Display the user's options and prompt for input...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun displayMainMenu(): Int {
    println()
    println("Task Options: ")
    println("(1) - Display All Data")
    println("(2) - Display a Record")
    println("(3) - Create a Record")
    println("(4) - Update a Record")
    println("(5) - Delete a Record")
    println("(6) - Exit the Program")
    print("Enter an Option Number: ")
    var num = -1
    try {
        num = readLine()!!.toInt()
    } catch (e: NumberFormatException) {
    }
    return num
}
```

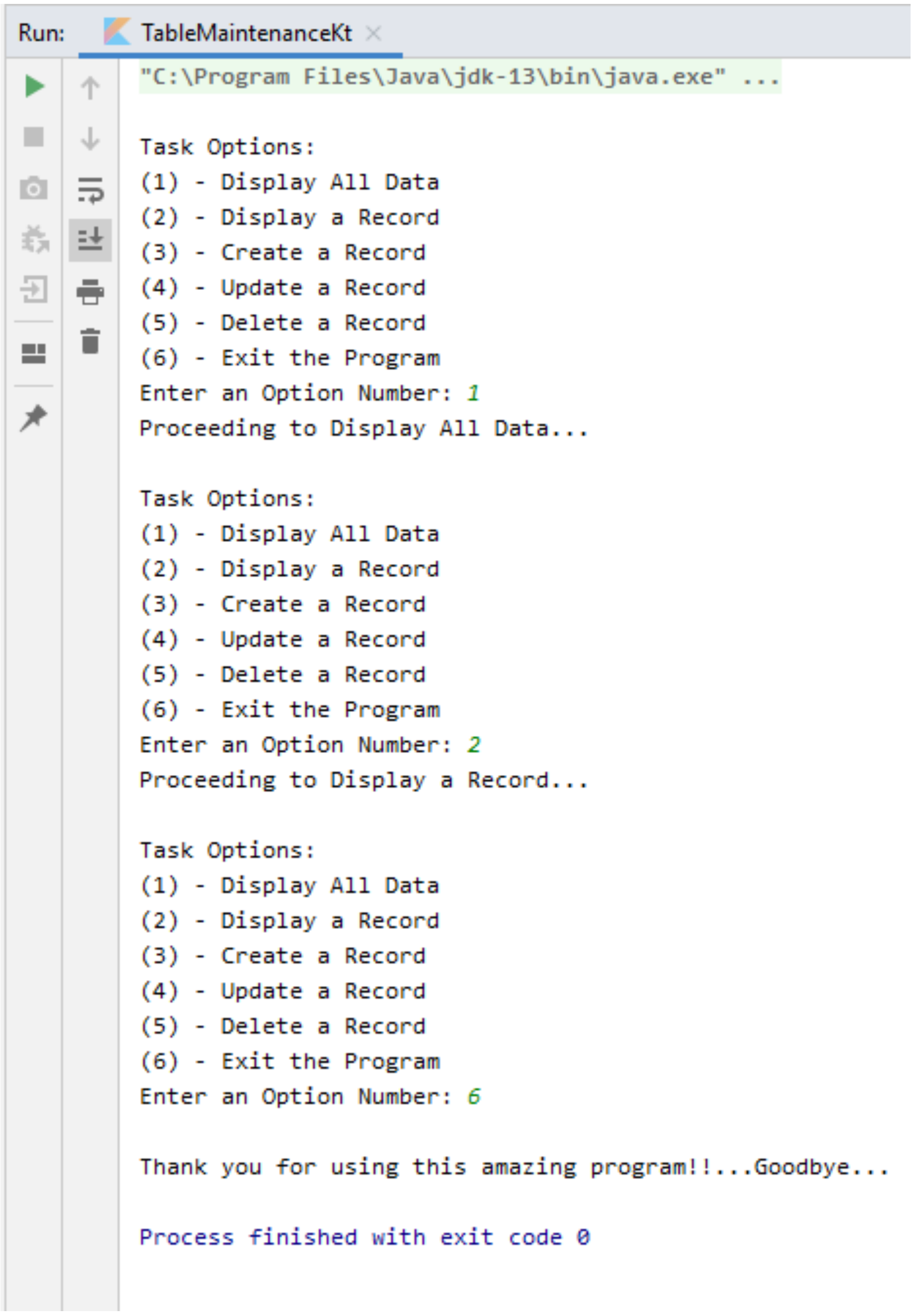
You may recall from our earlier program flow diagram, we want this to be in an infinite loop, until the user chooses to exit with Option 6:



Replace the main() method with the following code:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method main: Entry point for the program...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
fun main() {
    //Process User Selection...
    var answer = 0
    while (answer != 6) {
        answer = displayMainMenu()
        when (answer) {
            1 -> println("Proceeding to Display All Data...")
            2 -> println("Proceeding to Display a Record...")
            3 -> println("Proceeding to Create a Record...")
            4 -> println("Proceeding to Update a Record...")
            5 -> println("Proceeding to Delete a Record...")
            6 -> {
                println("\nThank you for using this amazing program!!...Goodbye...")
                exitProcess(0)
            }
            else -> {
                println()
                println("Invalid Option Number. Please Enter a valid option number 1 thru 6.")
            }
        }
    }
}
```

Execute the program, and you can test out the user menu. Hit Option 6 to exit:



```
"C:\Program Files\Java\jdk-13\bin\java.exe" ...

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 1
Proceeding to Display All Data...

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 2
Proceeding to Display a Record...

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 6

Thank you for using this amazing program!!...Goodbye...

Process finished with exit code 0
```

Now we have our infinite loop, until the user chooses to exit. From our earlier exercises to test the Gateway service, we know there are some strings we will need throughout the program. Create the following constants at the top of the program, above our main method. Be sure to use your own system (MAIN_URL) and auth string (AUTH_STRING), rather than the below:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Global Final Variables used throughout the program (val = Final, var = Variable)...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
const val AUTH_STRING = "amNhcHBblahblahblahblahc="
const val MAIN_URL = "http://<your server>/sap/opu/odata/SAP/ZTEST_KOTLIN_SRV/ZTEST_KOTLINSet"
const val JSON_FORMAT = "?\$format=json"

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

```kotlin
// Method main: Entry point for the program...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
fun main() {
```

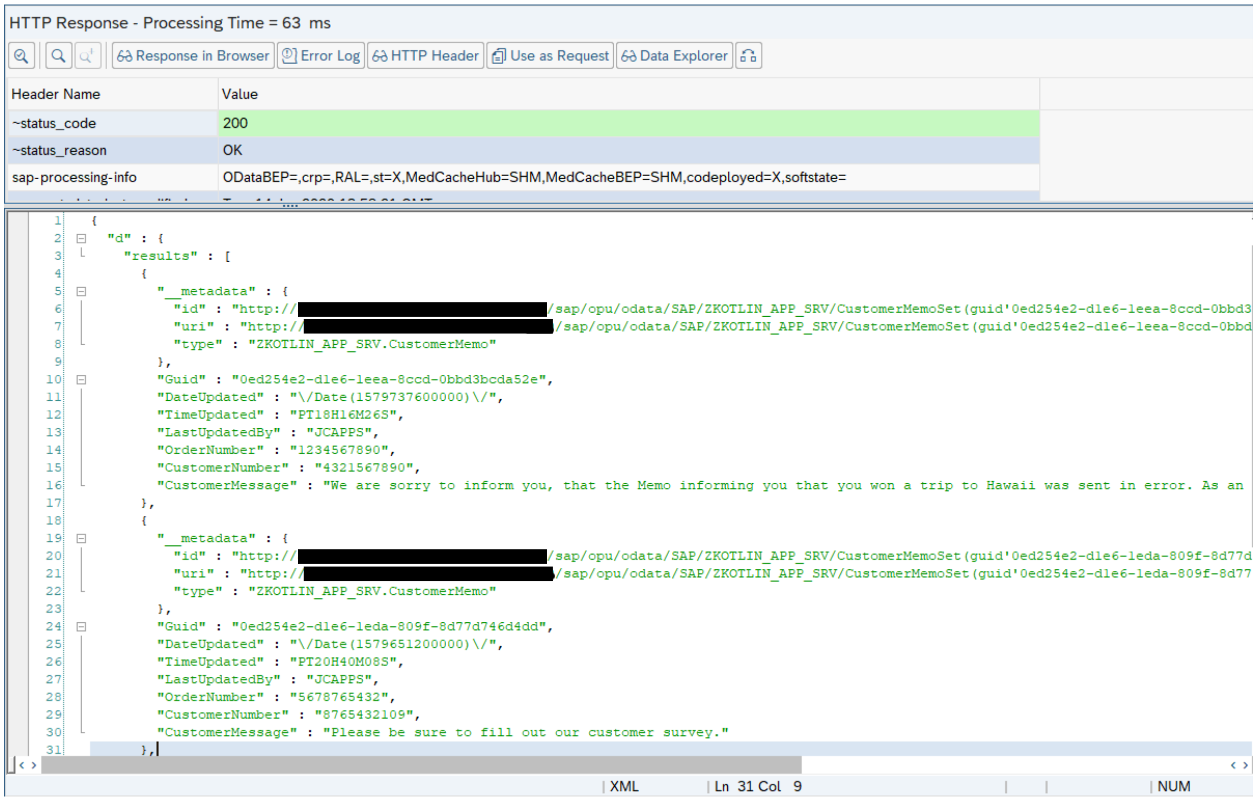
Before we write our next method, let's talk about the format of the JSON we will receive from the SAP Gateway, and how we may translate that into a Kotlin class…

DATA CLASSES

Kotlin has special classes that we can define only to hold data:

https://kotlinlang.org/docs/reference/data-classes.html

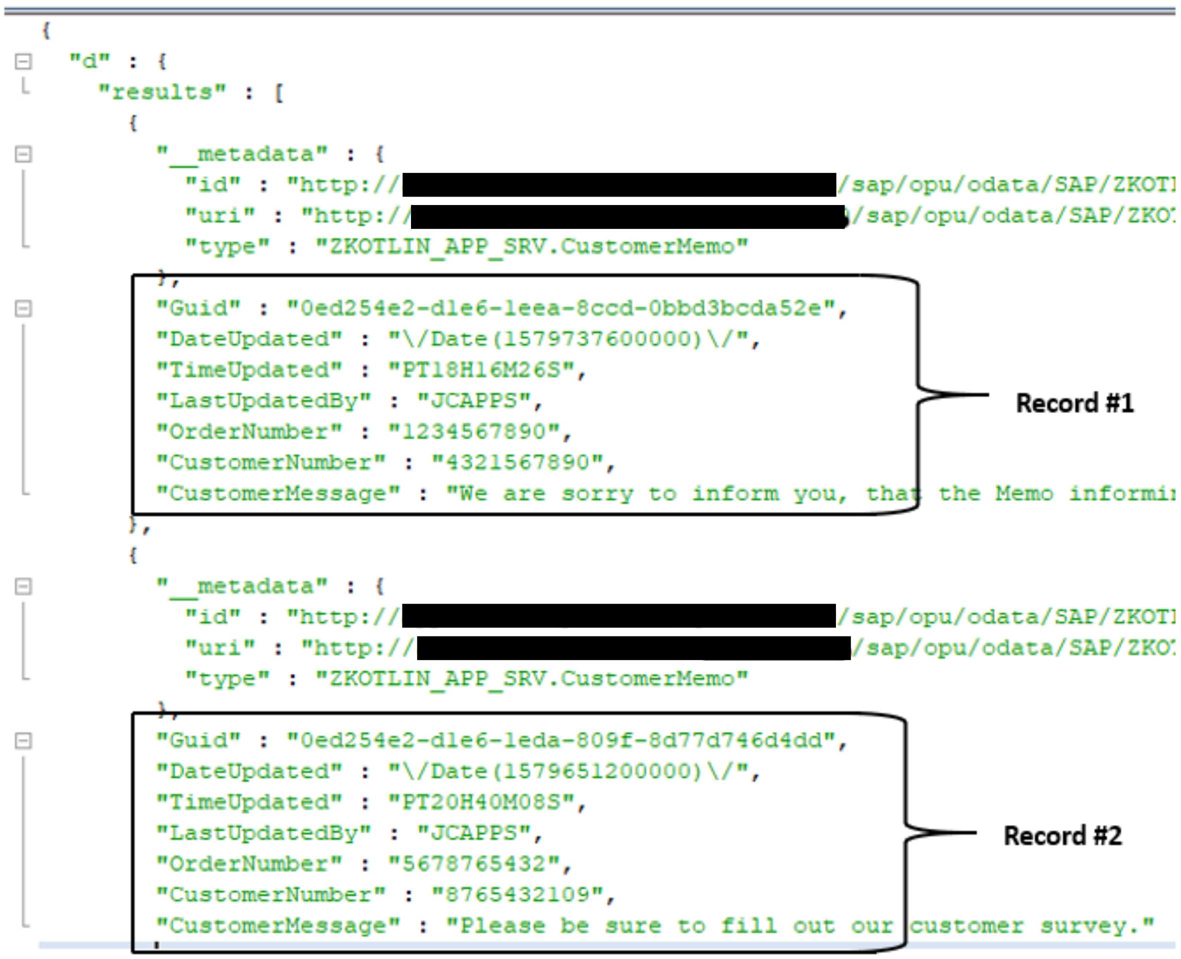
A perfect fit as a container to hold our JSON data nodes. Recall earlier, while testing our SAP Gateway service, we had the following JSON results:



Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=X,MedCacheHub=SHM,MedCacheBEP=SHM,codeployed=X,softstate=

```
{
  "d" : {
    "results" : [
      {
        "__metadata" : {
          "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8ccd-0bbd3
          "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eea-8ccd-0bbd
          "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
        },
        "Guid" : "0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e",
        "DateUpdated" : "\/Date(1579737600000)\/",
        "TimeUpdated" : "PT18H16M26S",
        "LastUpdatedBy" : "JCAPPS",
        "OrderNumber" : "1234567890",
        "CustomerNumber" : "4321567890",
        "CustomerMessage" : "We are sorry to inform you, that the Memo informing you that you won a trip to Hawaii was sent in error. As an
      },
      {
        "__metadata" : {
          "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77d
          "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254e2-d1e6-1eda-809f-8d77
          "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
        },
        "Guid" : "0ed254e2-d1e6-1eda-809f-8d77d746d4dd",
        "DateUpdated" : "\/Date(1579651200000)\/",
        "TimeUpdated" : "PT20H40M08S",
        "LastUpdatedBy" : "JCAPPS",
        "OrderNumber" : "5678765432",
        "CustomerNumber" : "8765432109",
        "CustomerMessage" : "Please be sure to fill out our customer survey."
      },
```

We have the main node of "d", then a sub node with "results". Within each repeating record, we have an "__metadata" node. We need to traverse this structure and fetch only the data we want. For example, to parse out the first 2 records, we want the following:



```
{
  "d" : {
    "results" : [
      {
        "__metadata" : {
          "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOT
          "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKO
          "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
        },
        "Guid" : "0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e",
        "DateUpdated" : "\/Date(1579737600000)\/",
        "TimeUpdated" : "PT18H16M26S",
        "LastUpdatedBy" : "JCAPPS",
        "OrderNumber" : "1234567890",
        "CustomerNumber" : "4321567890",
        "CustomerMessage" : "We are sorry to inform you, that the Memo informi
      },
      {
        "__metadata" : {
          "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOT
          "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKO
          "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
        },
        "Guid" : "0ed254e2-d1e6-1eda-809f-8d77d746d4dd",
        "DateUpdated" : "\/Date(1579651200000)\/",
        "TimeUpdated" : "PT20H40M08S",
        "LastUpdatedBy" : "JCAPPS",
        "OrderNumber" : "5678765432",
        "CustomerNumber" : "8765432109",
        "CustomerMessage" : "Please be sure to fill out our customer survey."
```

To “Deserialize” a JSON string is to convert it from the above string into a Kotlin object. To “Serialize”, is to go the reverse, from an object back into a string. We’ll need this throughout our program, and we’ll use the Jackson library to help us. We’ll be declaring the following Jackson mapper throughout the program:

```
val mapper = jacksonObjectMapper()
```

To represent only the record itself, we’ll define the following Kotlin Data Class:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// A data class which represents a single Customer Memo Record from the ZTEST_KOTLIN SAP table.
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
data class CustomerMemoRecord(
    var Guid: String = "",
    var DateUpdated: String = "",
    var TimeUpdated: String = "",
    var LastUpdatedBy: String = "",
    var OrderNumber: String = "",
    var CustomerNumber: String = "",
    var CustomerMessage: String = ""
)
```

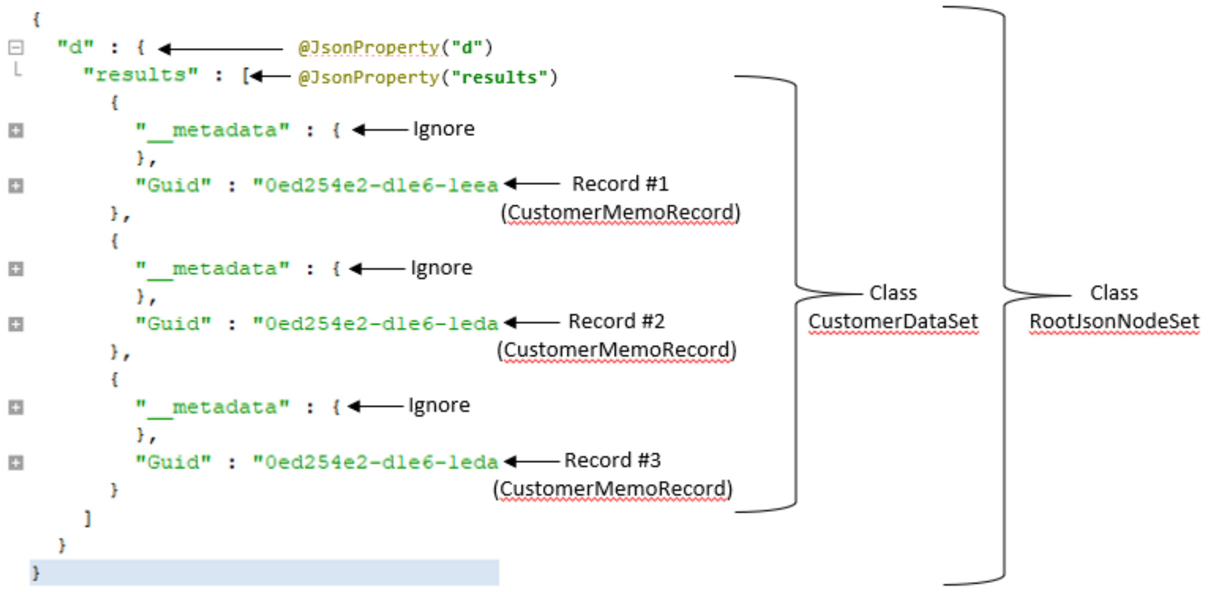
To mirror our JSON string into an object, to fetch multiple records, we’ll define a Kotlin “List”, which will represent all records:

```
val customerRecords: List<CustomerMemoRecord> = emptyList()
```

To mirror the JSON string into a complete set of records, and traverse the “d” and “results” nodes, we’ll define the following 2 Kotlin Data Classes:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Multiple Record GET - RootJsonNodeSet plus CustomerDataSet with all Customer Records...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
data class RootJsonNodeSet(
    @JsonProperty("d")
    val allRecords: CustomerDataSet
)
data class CustomerDataSet(
    @JsonProperty("results")
    val customerRecords: List<CustomerMemoRecord> = emptyList()
)
```

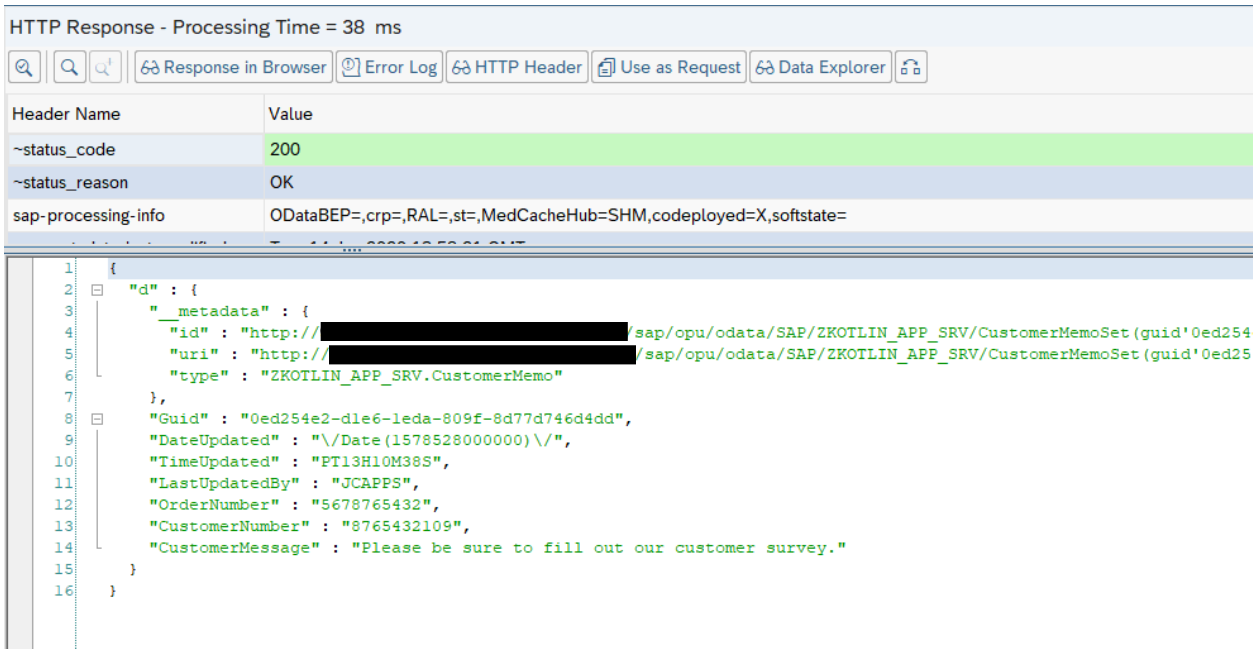
If we minimize the JSON Nodes, we can correlate more easily with the above Data Class definition:



```
{
  "d" : {
    "results" : [
      {
        "__metadata" : {
        },
        "Guid" : "0ed254e2-d1e6-1eea
      },
      {
        "__metadata" : {
        },
        "Guid" : "0ed254e2-d1e6-1eda
      },
      {
        "__metadata" : {
        },
        "Guid" : "0ed254e2-d1e6-1eda
      }
    ]
  }
}
```

The above will correspond to the CUSTOMERMEMOSET_GET_ENTITYSET method we coded earlier in our SAP Gateway service.

To fetch a single record, the JSON string looks a little different:



Header Name	Value
~status_code	200
~status_reason	OK
sap-processing-info	ODataBEP=,crp=,RAL=,st=,MedCacheHub=SHM,codeployed=X,softstate=

```
{
  "d" : {
    "__metadata" : {
      "id" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed254
      "uri" : "http://[redacted]/sap/opu/odata/SAP/ZKOTLIN_APP_SRV/CustomerMemoSet(guid'0ed25
      "type" : "ZKOTLIN_APP_SRV.CustomerMemo"
    },
    "Guid" : "0ed254e2-dle6-leda-809f-8d77d746d4dd",
    "DateUpdated" : "\/Date(1578528000000)\/",
    "TimeUpdated" : "PT13H10M38S",
    "LastUpdatedBy" : "JCAPPS",
    "OrderNumber" : "5678765432",
    "CustomerNumber" : "8765432109",
    "CustomerMessage" : "Please be sure to fill out our customer survey."
  }
}
```

Notice that the "results" node is not in a single record for an HTTP GET. For single records, we'll define the following Data Class:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Single Record GET for one Customer Memo Record...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
data class RootJsonNodeSingle(
    @JsonProperty("d")
    val singleCustomerRecord: CustomerMemoRecord
)
```

The above will correspond to the CUSTOMERMEMOSET_GET_ENTITY method we coded earlier in our SAP Gateway service, which fetches a single record.

You may be wondering how we ignore the "__metadata" node. The Jackson Object Mapper has a setting to ignore unknown properties that we don't have defined in our data class:

```
//Ignore any properties that don't exist in our CustomerMemoRecord Class...
mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
```

Let's define one last Data Class, which will represent a GET we will need to do for CUD (Create, Update, Delete) operations in order to fetch the CSRF Token and Cookie:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Data Class for a Fetch record, for CUD Operations...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
data class FetchRecord(
    val CsrfToken: String,
    val Cookie: String,
    var singleCustomerRecord: CustomerMemoRecord
)
```

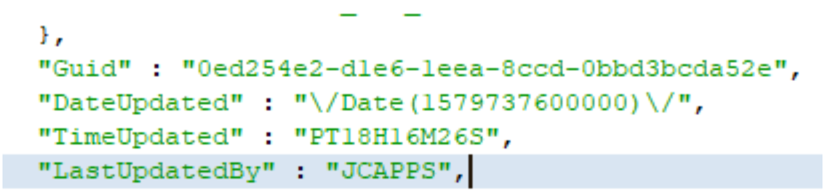
Add the 5 above Data Classes at the top of the program, just above the "fun main()" method. You may have noticed that IntelliJ has automatically added the following 2 imports to the top of our program, as we are working:

```
import com.fasterxml.jackson.annotation.JsonProperty
import kotlin.system.exitProcess
```

Other various library imports will be added by IntelliJ as we continue to write code.

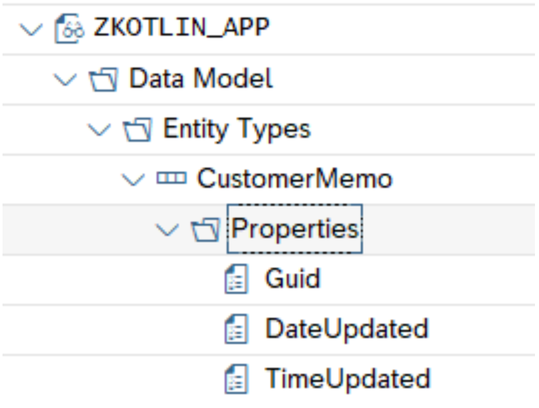
DATE AND TIME FORMATS

You may have noticed the strange formats for Date and Time Data Types:

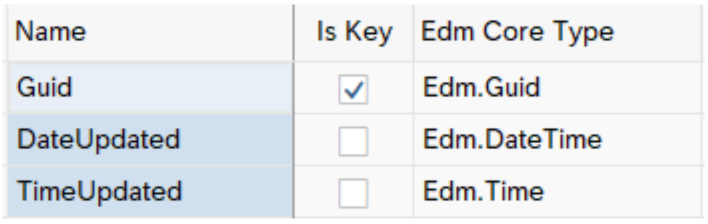


```
  },
  "Guid" : "0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e",
  "DateUpdated" : "\/Date(1579737600000)\/",
  "TimeUpdated" : "PT18H16M26S",
  "LastUpdatedBy" : "JCAPPS",
```

SAP Gateway Definition:



Double-click on Properties, to see the definition:



Name	Is Key	Edm Core Type
Guid	✓	Edm.Guid
DateUpdated	☐	Edm.DateTime
TimeUpdated	☐	Edm.Time

Edm.DateTime is an Epoch Date, which is a computer time that represents the amount of seconds that have passed since a point in time.

Edm.Time is an XSD:Duration Date Type.

For detailed links with information about these formats, see the Reference section below and the section titled “SAP Gateway Date and Time Format:”.

Add the following 2 methods to the bottom of our program, which will convert these Dates and Times into something readable:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Method jsonDateFormatter: Parse a JSON Date (Epoch Date) into a Java/Kotlin Date
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
private fun jsonDateFormatter(jsonDate: String): String {
    val epochDate = jsonDate.replace("[^0-9]".toRegex(), "")
    val updateDate = Instant.ofEpochMilli(java.lang.Long.parseLong(epochDate))
        .atZone(ZoneId.of("CET"))
        .toLocalDate()
    return updateDate.toString()
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Method jsonTimeFormatter: Parse a JSON Time (XSD:Duration Date Type) into Java/Kotlin Time
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
private fun jsonTimeFormatter(jsonTime: String): String {
    val fTime = LocalTime.ofNanoOfDay(Duration.parse(jsonTime).toNanos())
    val df = DateTimeFormatter.ISO_LOCAL_TIME
    return fTime.format(df)
}
```

The above 2 methods will convert these date and time formats into the following:

Date: YYYY-MM-DD, for example 2020-01-23
Time: HH:MM:SS, for example 18:16:26

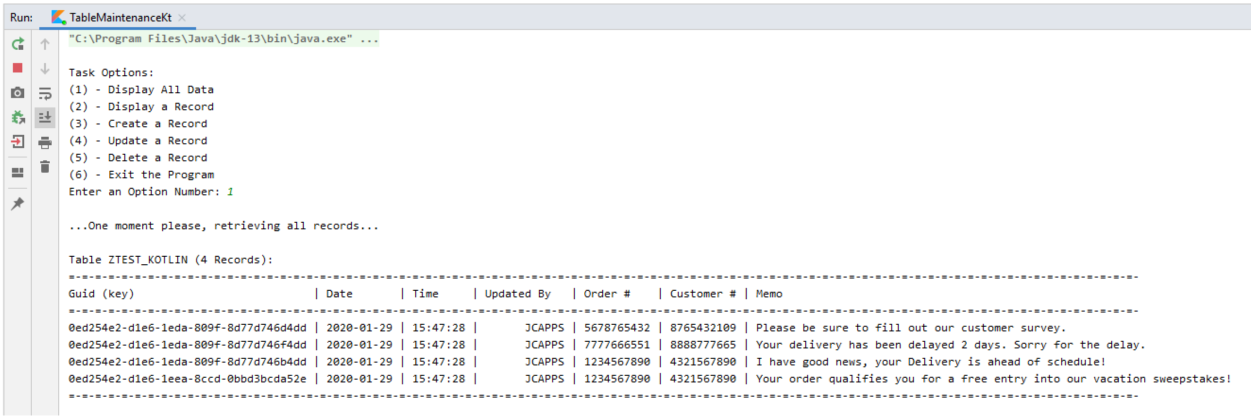
Next, let's write the code to perform an HTTP GET to read one or more records and call our SAP Gateway service. Create the following method to display all records:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method displayDataSet: Display all records in the ZTEST_KOTLIN SAP table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun displayDataSet() {
    println()
    println("...One moment please, retrieving all records...")
    FuelManager.instance.baseHeaders = mapOf("Authorization" to "Basic $AUTH_STRING")
    val mapper = jacksonObjectMapper()
    val url = MAIN_URL + JSON_FORMAT
    //This allows you to parse out only the attributes you'd like, and ignore all others...
    mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
    val (_, _, result) = url.httpGet().responseString()
    when (result) {
        is Result.Failure -> {
            val ex = result.getException()
            print(ex)
        }
        is Result.Success -> {
            val myJsonData: RootJsonNodeSet = mapper.readValue(result.get())
            val memoRecords: List<CustomerMemoRecord> = myJsonData.allRecords.customerRecords
            println("\nTable ZTEST_KOTLIN (${memoRecords.count()} Records):")
            for (x in 0..80) print("=-") // Print 80 times for line separator...
            println()
            println("Guid (key)                          " +
                    "| Date       " +
                    "| Time     " +
                    "| Updated By   " +
                    "| Order #    " +
                    "| Customer # " +
                    "| Memo")
            for (x in 0..80) print("=-") // Print 80 times...
            println()
            memoRecords.forEach {
                println(
                    "${it.Guid} " +
                            "| ${jsonDateFormatter(it.DateUpdated)} " +
                            "| ${jsonTimeFormatter(it.TimeUpdated)} " +
                            "| ${it.LastUpdatedBy.padStart(12)} " + //pad to 12 characters, to
line up with column header
                            "| ${it.OrderNumber.padStart(10)} " + //pad to 10 characters
                            "| ${it.CustomerNumber.padStart(10)} " + //pad to 10 characters
                            "| ${it.CustomerMessage}"
                )
            }
            for (x in 0..80) print("=-") // Print 80 times...
            println()
        }
    }
}
```

Next, we need to update our user menu in the main method, so it calls the above displayDataSet() method:

```
when (answer) {
    1 -> displayDataSet() //<<<add this
```

Execute the program (run the main method), and select Option 1 in the user menu:



```
"C:\Program Files\Java\jdk-13\bin\java.exe" ...

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 1

...One moment please, retrieving all records...

Table ZTEST_KOTLIN (4 Records):
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Guid (key)                           | Date       | Time     | Updated By   | Order #    | Customer # | Memo
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
0ed254e2-d1e6-1eda-809f-8d77d746d4dd | 2020-01-29 | 15:47:28 |       JCAPPS | 5678765432 | 8765432109 | Please be sure to fill out our customer survey.
0ed254e2-d1e6-1eda-809f-8d77d746f4dd | 2020-01-29 | 15:47:28 |       JCAPPS | 7777666551 | 8888777665 | Your delivery has been delayed 2 days. Sorry for the delay.
0ed254e2-d1e6-1eda-809f-8d77d746b4dd | 2020-01-29 | 15:47:28 |       JCAPPS | 1234567890 | 4321567890 | I have good news, your Delivery is ahead of schedule!
0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e | 2020-01-29 | 15:47:28 |       JCAPPS | 1234567890 | 4321567890 | Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

The above displayed all records currently in the ZTEST_KOTLIN table.

To read a single record, we will require the user to enter the Guid Key for the record. Create the following method, for prompting the user. We will use this method in several places, so we'll create a generic method that simply prompts the user with a message and returns their entry:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method promptTheUser: Prompt the end user for something...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun promptTheUser(message: String): String {
    print(message)
    return readLine()!!
}
```

Because we will also need to perform an HTTP GET to retrieve our tokens for the CUD operations, let's create a generic method that simply fetches a record:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method fetchSingleRecord: Fetch Data for a Single Record...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun fetchSingleRecord(lclUrl: String): FetchRecord {
    val mapper = jacksonObjectMapper()
    FuelManager.instance.baseHeaders = mapOf(
        "Authorization" to "Basic $AUTH_STRING",
        "X-CSRF-Token" to "Fetch"
    )
    //Ignore any properties that don't exist in our CustomerMemoRecord Class...
    mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
    val (_, response, result) = lclUrl.httpGet().responseString()
    val csrfToken: String? = response.headers["x-csrf-token"].elementAt(0)
    val cookie: String? = response.headers["set-cookie"].elementAt(0) + response.headers["set-
cookie"].elementAt(1)
    var memoRecord = CustomerMemoRecord()
    when (result) {
        is Result.Failure -> {
            val ex = result.getException()
            print(ex)
        }
        is Result.Success -> {
            val myJsonData: RootJsonNodeSingle = mapper.readValue(result.get())
            memoRecord = myJsonData.singleCustomerRecord
        }
    }
    return FetchRecord(
        CsrfToken = csrfToken.toString(),
        Cookie = cookie.toString(),
        singleCustomerRecord = memoRecord
    )
}
```

Also, because there will be multiple places we will want to display a single record (for example, after we update it), lets create a generic method which formats a single record:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method showFormattedRecord: Display of a single record...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun showFormattedRecord(record: CustomerMemoRecord, lclTitle: String) {
    println()
    for (x in 0..50) print("=-") // Print 50 times for line separator...
    println()
    println("$lclTitle Record for Order Number ${record.OrderNumber}:")
    for (x in 0..50) print("=-") // Print 50 times...
    println()
    println("          Guid (key): ${record.Guid}")
    println("   Date Last Updated: ${jsonDateFormatter(record.DateUpdated)}")
    println("   Time Last Updated: ${jsonTimeFormatter(record.TimeUpdated)}")
    println("Last Updated By User: ${record.LastUpdatedBy}")
    println("        Order Number: ${record.OrderNumber}")
    println("     Customer Number: ${record.CustomerNumber}")
    println("       Customer Memo: ${record.CustomerMessage}")
    for (x in 0..50) print("=-") // Print 50 times...
    println()
}
```

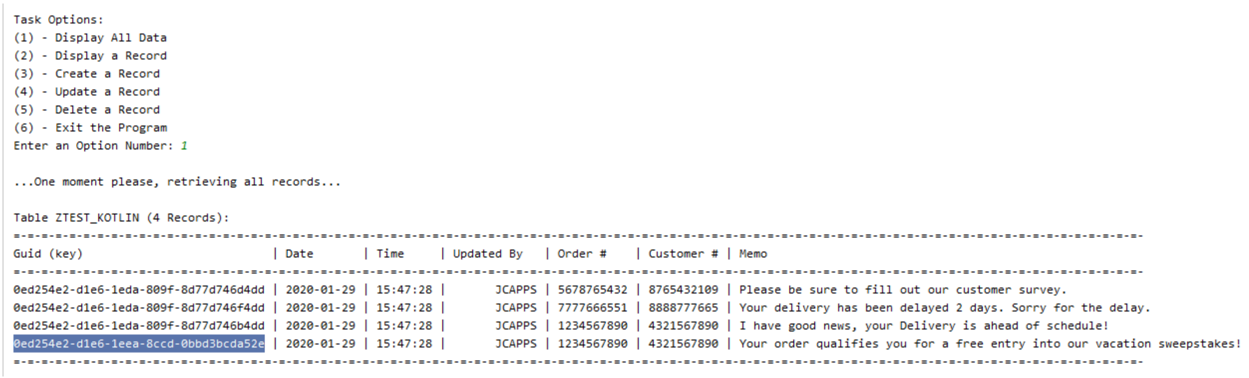
Finally, here is the method which displays a single record:

```kotlin
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Method displaySingleRecord: Display a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
private fun displayRecord(lclGuid: String, lclTitle: String) {
    val singleRecordString = "(Guid=guid'${lclGuid}')"
    val lclUrl = MAIN_URL + singleRecordString + JSON_FORMAT
    val fetchRecord = fetchSingleRecord(lclUrl)
    showFormattedRecord( fetchRecord.singleCustomerRecord, lclTitle )
}
```

Lastly, we need to call our new method from the user menu. Add the following to the main() method for user menu option "2":

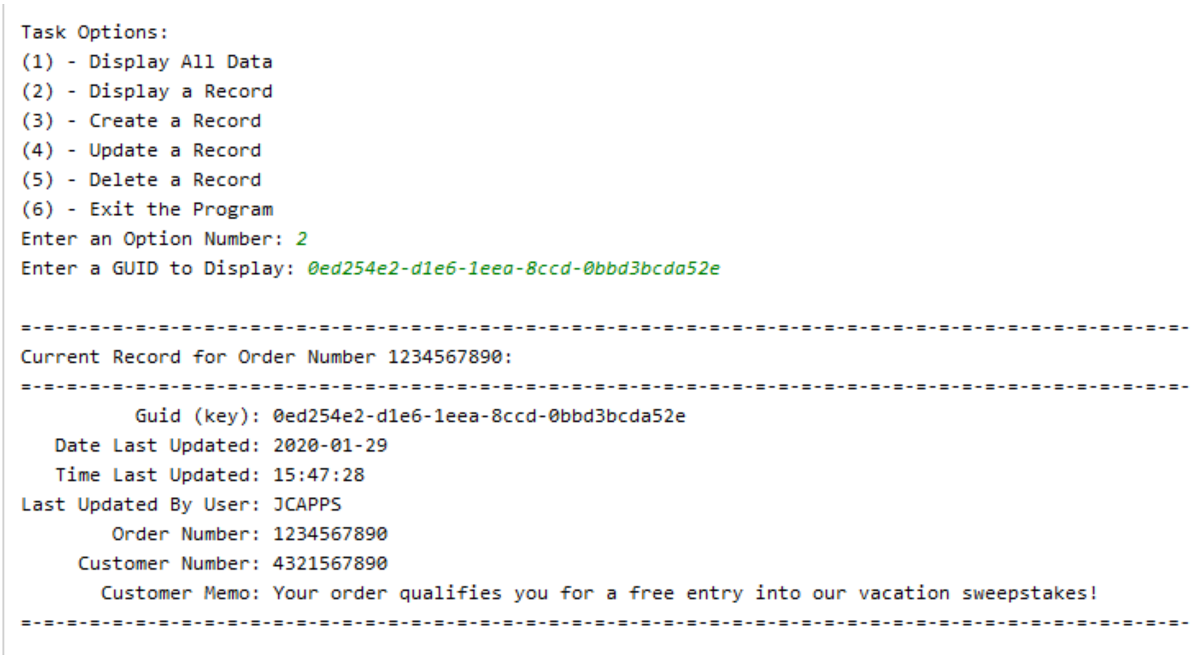
```kotlin
when (answer) {
    1 -> displayDataSet()
    2 -> displayRecord(promptTheUser("Enter a GUID to Display: "), "Current")
```

In order to get a Guid Key, you can run option 1, first, to display all records, then copy and paste the Guid in order to run option 2 and display the single record. For example, you can run it and do the following:



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 1

...One moment please, retrieving all records...

Table ZTEST_KOTLIN (4 Records):
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Guid (key)                           | Date       | Time     | Updated By | Order #    | Customer # | Memo
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
0ed254e2-d1e6-1eda-809f-8d77d746d4dd | 2020-01-29 | 15:47:28 |     JCAPPS | 5678765432 | 8765432109 | Please be sure to fill out our customer survey.
0ed254e2-d1e6-1eda-809f-8d77d746f4dd | 2020-01-29 | 15:47:28 |     JCAPPS | 7777666551 | 8888777665 | Your delivery has been delayed 2 days. Sorry for the delay.
0ed254e2-d1e6-1eda-809f-8d77d746b4dd | 2020-01-29 | 15:47:28 |     JCAPPS | 1234567890 | 4321567890 | I have good news, your Delivery is ahead of schedule!
0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e | 2020-01-29 | 15:47:28 |     JCAPPS | 1234567890 | 4321567890 | Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Highlight and Copy the last Guid (key), then run with option 2:



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 2
Enter a GUID to Display: 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Current Record for Order Number 1234567890:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e
   Date Last Updated: 2020-01-29
   Time Last Updated: 15:47:28
Last Updated By User: JCAPPS
        Order Number: 1234567890
     Customer Number: 4321567890
       Customer Memo: Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

CREATE RECORD

Now, we've completed all of our HTTP GET methods. Let's move on to the remainder of our methods, which will Create, Update and Delete records.

This is the first operation where we will be performing an actual SAP Database update from our Kotlin application. Earlier in the SAP Gateway setup and testing section, we explained how to setup a "GET" operation to fetch our CSRF Token and Cookie for security required when updating the database. For read only operations (i.e. GET), we don't need these tokens. Above, we created a data class called "FetchRecord" to hold our tokens, and wrote a method called "FetchSingleRecord" which will store our tokens.

A few items we will need for our Create Record method:

1. Data required from the user to create a new record is:
   - Order Number
   - Customer Number
   - Customer Memo

   So we will prompt the user 3 times by calling our generic "promptTheUser" method.
2. Call our Gateway Service for an HTTP GET, with our token "Fetch" option, to get the necessary tokens to update the backend database.
3. Fill the JSON string with the Order Number, Customer Number and Customer Memo, which the user entered in step 1 using our Jackson Object Mapper.
4. Call the SAP Gateway service and do an HTTP POST with our JSON string to create the new record.
5. Display the newly created record to the end-user, along with the new Guid Key and other administrative fields (Date Created, Time Created, etc.).

First, there is a bit of common code we will be using each time we do a Create, Update and Delete. Let's write the following reusable method, which will initialize the headers for our Fuel HTTP Library:

```kotlin
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method initializeFuel: Set the FuelManager for UPDATE or POST...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun initializeFuel(CsrfToken: String, Cookie: String) {
    FuelManager.instance.baseHeaders = mapOf(
        "Authorization" to "Basic $AUTH_STRING",
        "X-CSRF-Token" to CsrfToken,
        "cookie" to Cookie,
        "Content-Type" to "application/json",
        "Accept" to "application/json"
    )
}
```

Create the following method called "createRecord":

```kotlin
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method createRecord: Create a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun createRecord() {
    val lclOrder = promptTheUser("Enter Order Number: ")
    val lclCustomer = promptTheUser("Enter Customer Number: ")
    val lclMemo = promptTheUser("Enter Customer Memo: ")

    //First, fetch the CSRF Token and Cookie, prior to performing the POST...
    var url = MAIN_URL + JSON_FORMAT
    val fetchRecord = fetchSingleRecord(url)
    fetchRecord.singleCustomerRecord.OrderNumber = lclOrder
    fetchRecord.singleCustomerRecord.CustomerNumber = lclCustomer
    fetchRecord.singleCustomerRecord.CustomerMessage = lclMemo
    //Even though we are doing a POST (Create), we still need to fill in all of the
    //attributes, so enter dummy data for these ignored fields...
    fetchRecord.singleCustomerRecord.Guid = "00000000-0000-0000-0000-000000000000"
    fetchRecord.singleCustomerRecord.DateUpdated = """/Date(1578441600000)/"""
    fetchRecord.singleCustomerRecord.LastUpdatedBy = ""
    fetchRecord.singleCustomerRecord.TimeUpdated = "PT13H12M09S"
    val mapper = jacksonObjectMapper()
    // The default mapper, adjusts the field names to lower case camel case, but our
    // Gateway service has upper case (i.e. dateUpdated vs. DateUpdated),
    // so we set the UPPER_CAMEL_CASE property here...
    mapper.propertyNamingStrategy = PropertyNamingStrategy.UPPER_CAMEL_CASE
    mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
    // Serialize the fetchRecord Object back into a JSON String and use for our POST
    // to create the Customer Memo...
    val jsonString = mapper.writeValueAsString(fetchRecord.singleCustomerRecord)
    //Remove the "jsonFormat" URI Option, prior to doing the POST...
    url = MAIN_URL
    val newRecord: CustomerMemoRecord

    initializeFuel(fetchRecord.CsrfToken, fetchRecord.Cookie)
    val postString = """{ "d" : $jsonString }"""

    //This is a synchronous "Blocking Mode" call (i.e. will wait for a response)...
    val (_, _, result) = url.httpPost().body(postString).responseString()
    when (result) {
        is Result.Failure -> {
            println()
            println("Post Failed...")
            println(result.getException().toString() + result.error.response.toString())
        }
        is Result.Success -> {
            val myJsonData: RootJsonNodeSingle = mapper.readValue(result.get())
            newRecord = myJsonData.singleCustomerRecord
            println()
            println("...Customer Memo successfully created...")
            displayRecord(newRecord.Guid, "New")
        }
    }
}
```

Lastly, add a call to our new “Create” method on our user menu for option 3:

```
when (answer) {
    1 -> displayDataSet()
    2 -> displayRecord(promptTheUser("Enter a GUID to Display: "), "Current")
    3 -> createRecord()
```

Run the program, and test the new Create Method:

```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 3
Enter Order Number: 4444
Enter Customer Number: 7777
Enter Customer Memo: This is my first Customer Memo Created from Kotlin!

...Customer Memo successfully created...

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
New Record for Order Number 4444:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-91db-7221d71bc7f8
   Date Last Updated: 2020-02-03
   Time Last Updated: 18:19:03
Last Updated By User: JCAPPS
        Order Number: 4444
     Customer Number: 7777
       Customer Memo: This is my first Customer Memo Created from Kotlin!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

UPDATE RECORD

An Update operation is very similar to our Create operation, above, except we will do an "HTTP PUT", to update an existing record, rather than an HTTP POST. Our Update Record method will do the following:

1. Prompt the user for a Guid Key for the record to be updated.
2. Prompt the user for the new Customer Memo to overwrite their existing Memo.
3. Call our fetchSingleRecord method for the Guid key to be updated. This will retrieve our tokens, plus a JSON string which represents the record.
4. Modify the above JSON String, and overwrite the memo with our new memo. Utilizing our Jackson Object Mapper, we simply update the CustomerMessage attribute in our CustomerMemoRecord object.
5. Perform an HTTP PUT with our Fuel HTTP library. Display the old Memo and the new Memo.
6. Display the newly updated record to the end user.

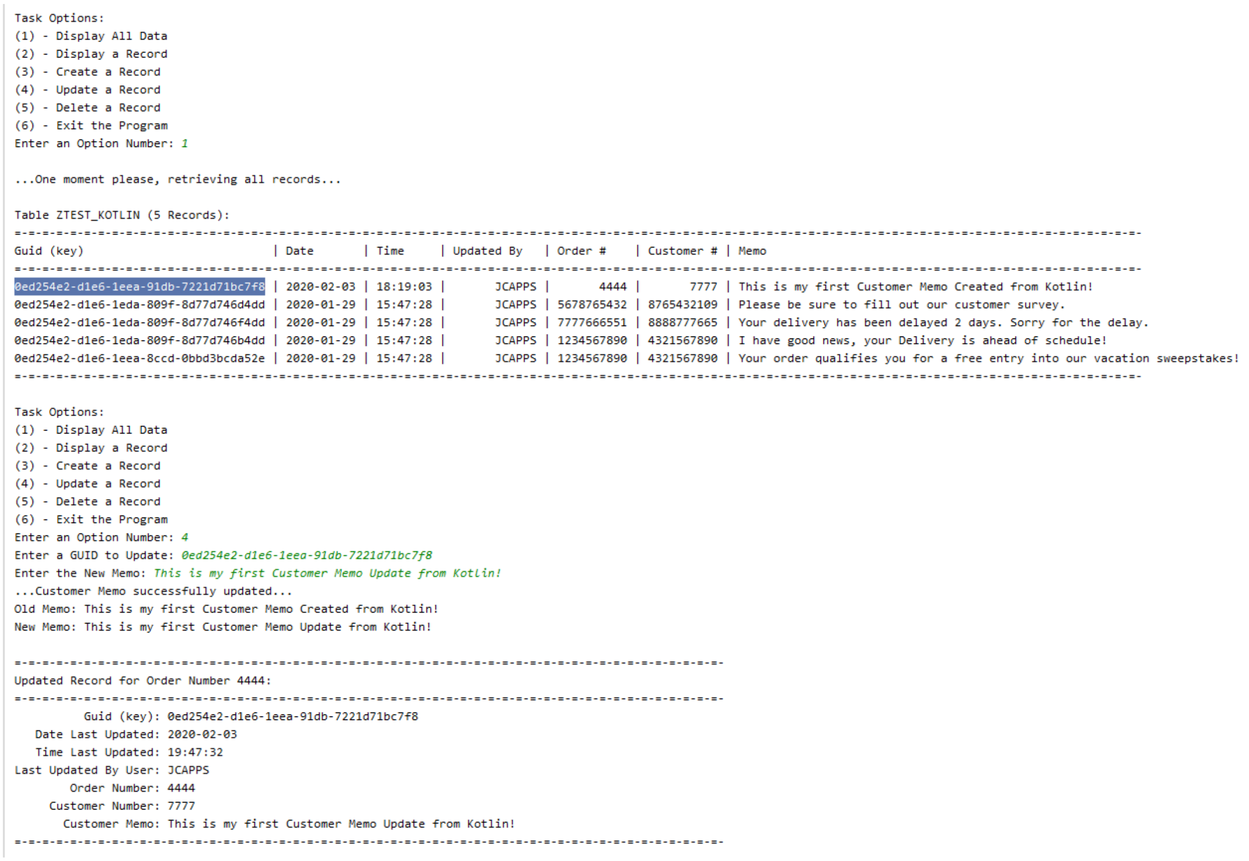
Create the following updateRecord method:

```kotlin
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method updateRecord: Update a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun updateRecord() {
    val lclGuid = promptTheUser("Enter a GUID to Update: ")
    val newMemo = promptTheUser("Enter the New Memo: ")
    var url = "$MAIN_URL(Guid=guid'$lclGuid')$JSON_FORMAT"
    val fetchRecord = fetchSingleRecord(url)
    val originalMemo = fetchRecord.singleCustomerRecord.CustomerMessage
    fetchRecord.singleCustomerRecord.CustomerMessage = newMemo
    val mapper = jacksonObjectMapper()
    mapper.propertyNamingStrategy = PropertyNamingStrategy.UPPER_CAMEL_CASE
    val jsonString = mapper.writeValueAsString(fetchRecord.singleCustomerRecord)
    url = "$MAIN_URL(Guid=guid'$lclGuid')"

    initializeFuel(fetchRecord.CsrfToken, fetchRecord.Cookie)
    val putString = """{ "d" : $jsonString }"""
    val (_, _, result) = url.httpPut().body(putString).responseString()
    when (result) {
        is Result.Failure -> {
            println(result.getException().toString())
        }
        is Result.Success -> {
            println("...Customer Memo successfully updated...")
            println("Old Memo: $originalMemo")
            println("New Memo: $newMemo")
            displayRecord(lclGuid, "Updated")
        }
    }
}
```

Add a call to our new "Update" method on our user menu for option 4:

```kotlin
when (answer) {
    1 -> displayDataSet()
    2 -> displayRecord(promptTheUser("Enter a GUID to Display: "), "Current")
    3 -> createRecord()
    4 -> updateRecord()
```

To test, we'll need to get a Guid Key for the record we'd like to update. You can display all records, with option 1, then copy a Guid key to input into the update record option 4, as follows:



```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 1

...One moment please, retrieving all records...

Table ZTEST_KOTLIN (5 Records):
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Guid (key)                           | Date       | Time     | Updated By | Order #    | Customer # | Memo
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
0ed254e2-d1e6-1eea-91db-7221d71bc7f8 | 2020-02-03 | 18:19:03 |     JCAPPS |       4444 |       7777 | This is my first Customer Memo Created from Kotlin!
0ed254e2-d1e6-1eda-809f-8d77d746d4dd | 2020-01-29 | 15:47:28 |     JCAPPS | 5678765432 | 8765432109 | Please be sure to fill out our customer survey.
0ed254e2-d1e6-1eda-809f-8d77d746f4dd | 2020-01-29 | 15:47:28 |     JCAPPS | 7777666551 | 8888777665 | Your delivery has been delayed 2 days. Sorry for the delay.
0ed254e2-d1e6-1eda-809f-8d77d746b4dd | 2020-01-29 | 15:47:28 |     JCAPPS | 1234567890 | 4321567890 | I have good news, your Delivery is ahead of schedule!
0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e | 2020-01-29 | 15:47:28 |     JCAPPS | 1234567890 | 4321567890 | Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 4
Enter a GUID to Update: 0ed254e2-d1e6-1eea-91db-7221d71bc7f8
Enter the New Memo: This is my first Customer Memo Update from Kotlin!
...Customer Memo successfully updated...
Old Memo: This is my first Customer Memo Created from Kotlin!
New Memo: This is my first Customer Memo Update from Kotlin!

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Updated Record for Order Number 4444:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-91db-7221d71bc7f8
   Date Last Updated: 2020-02-03
   Time Last Updated: 19:47:32
Last Updated By User: JCAPPS
        Order Number: 4444
     Customer Number: 7777
       Customer Memo: This is my first Customer Memo Update from Kotlin!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

DELETE RECORD

The Deletion method is a bit simpler. Since we are simply deleting the record, we only need to provide the Guid Key. We still need to fetch the tokens, just as we did with Create and Update operations.

Our Delete Method will perform the following:

1. Prompt the user for a Guid Key for the record to be deleted.
2. Call our fetchSingleRecord method for the Guid key to be deleted and retrieve our tokens.
3. Perform an HTTP DELETE with our Fuel HTTP library, adding the Guid to the URL.
4. Display the deleted record to the end user.

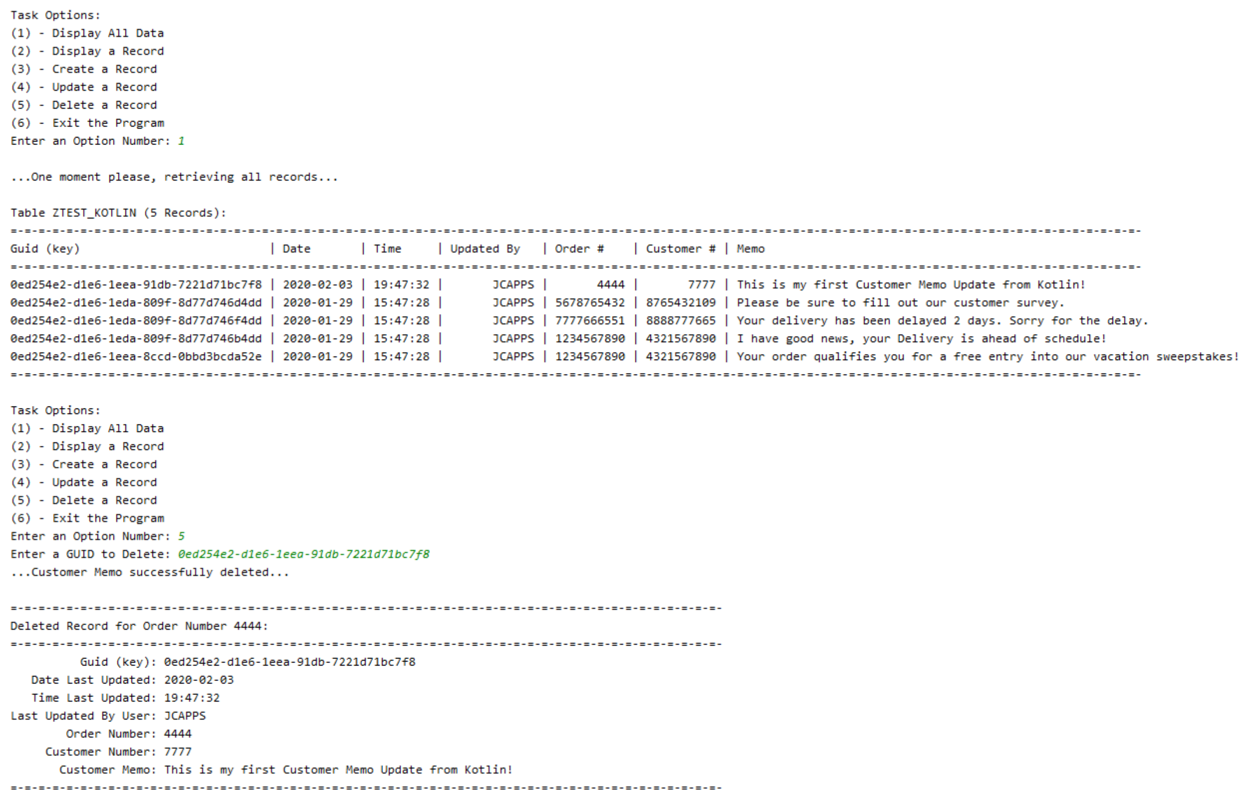
Create the following deleteRecord method:

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method deleteRecord: Delete a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun deleteRecord() {
    val lclGuid = promptTheUser("Enter a GUID to Delete: ")
    var url = "$MAIN_URL(Guid=guid'$lclGuid')$JSON_FORMAT"
    val fetchRecord = fetchSingleRecord(url)
    val mapper = jacksonObjectMapper()
    mapper.propertyNamingStrategy = PropertyNamingStrategy.UPPER_CAMEL_CASE
    url = "$MAIN_URL(Guid=guid'$lclGuid')"
    initializeFuel(fetchRecord.CsrfToken, fetchRecord.Cookie)
    val (_, _, result) = url.httpDelete().responseString()
    when (result) {
        is Result.Failure -> {
            println(result.getException().toString())
        }
        is Result.Success -> {
            println("...Customer Memo successfully deleted...")
            showFormattedRecord( fetchRecord.singleCustomerRecord, "Deleted" )
        }
    }
}
```

Add a call to our new “Delete” method on our user menu for option 5:

```
when (answer) {
    1 -> displayDataSet()
    2 -> displayRecord(promptTheUser("Enter a GUID to Display: "), "Current")
    3 -> createRecord()
    4 -> updateRecord()
    5 -> deleteRecord()
```

To test, display all records with option 1, decide on which record you want to delete, and copy the Guid key. Run option 5 to delete the record by pasting in the Guid key. You can test as follows:


```
Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 1

...One moment please, retrieving all records...

Table ZTEST_KOTLIN (5 Records):
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Guid (key)                           | Date       | Time     | Updated By   | Order #    | Customer # | Memo
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
0ed254e2-d1e6-1eea-91db-7221d71bc7f8 | 2020-02-03 | 19:47:32 |       JCAPPS |       4444 |       7777 | This is my first Customer Memo Update from Kotlin!
0ed254e2-d1e6-1eda-809f-8d77d746d4dd | 2020-01-29 | 15:47:28 |       JCAPPS | 5678765432 | 8765432109 | Please be sure to fill out our customer survey.
0ed254e2-d1e6-1eda-809f-8d77d746f4dd | 2020-01-29 | 15:47:28 |       JCAPPS | 7777666551 | 8888777665 | Your delivery has been delayed 2 days. Sorry for the delay.
0ed254e2-d1e6-1eda-809f-8d77d746b4dd | 2020-01-29 | 15:47:28 |       JCAPPS | 1234567890 | 4321567890 | I have good news, your Delivery is ahead of schedule!
0ed254e2-d1e6-1eea-8ccd-0bbd3bcda52e | 2020-01-29 | 15:47:28 |       JCAPPS | 1234567890 | 4321567890 | Your order qualifies you for a free entry into our vacation sweepstakes!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Task Options:
(1) - Display All Data
(2) - Display a Record
(3) - Create a Record
(4) - Update a Record
(5) - Delete a Record
(6) - Exit the Program
Enter an Option Number: 5
Enter a GUID to Delete: 0ed254e2-d1e6-1eea-91db-7221d71bc7f8
...Customer Memo successfully deleted...

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Deleted Record for Order Number 4444:
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
          Guid (key): 0ed254e2-d1e6-1eea-91db-7221d71bc7f8
   Date Last Updated: 2020-02-03
   Time Last Updated: 19:47:32
Last Updated By User: JCAPPS
        Order Number: 4444
     Customer Number: 7777
       Customer Memo: This is my first Customer Memo Update from Kotlin!
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

COMPLETE KOTLIN PROGRAM

Here is the complete source code for the single TableMaintenance.kt Kotlin file:

```kotlin
import com.fasterxml.jackson.annotation.JsonProperty
import com.fasterxml.jackson.databind.DeserializationFeature
import com.fasterxml.jackson.databind.PropertyNamingStrategy
import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
import com.fasterxml.jackson.module.kotlin.readValue
import com.github.kittinunf.fuel.core.FuelManager
import com.github.kittinunf.fuel.httpDelete
import com.github.kittinunf.fuel.httpGet
import com.github.kittinunf.fuel.httpPost
import com.github.kittinunf.fuel.httpPut
import com.github.kittinunf.result.Result
import java.time.Duration
import java.time.Instant
import java.time.LocalTime
import java.time.ZoneId
import java.time.format.DateTimeFormatter
import kotlin.system.exitProcess

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Global Final Variables used throughout the program (val = Final, var = Variable)...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
const val AUTH_STRING = "amNhc<blah><blah><blah>FSUkVMTDc="
const val MAIN_URL = "http://<your server>/sap/opu/odata/SAP/ZTEST_KOTLIN_SRV/ZTEST_KOTLINSet"
const val JSON_FORMAT = "?\$format=json"

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Data Class for a Fetch record, for CUD Operations...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
data class FetchRecord(
    val CsrfToken: String,
    val Cookie: String,
    var singleCustomerRecord: CustomerMemoRecord
)

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// A data class which represents a single Customer Memo Record from the ZTEST_KOTLIN SAP table.
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
data class CustomerMemoRecord(
    var Guid: String = "",
    var DateUpdated: String = "",
    var TimeUpdated: String = "",
    var LastUpdatedBy: String = "",
    var OrderNumber: String = "",
    var CustomerNumber: String = "",
    var CustomerMessage: String = ""
)

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Multiple Record GET - RootJsonNodeSet plus CustomerDataSet with all Customer Records...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
data class RootJsonNodeSet(
    @JsonProperty("d")
    val allRecords: CustomerDataSet
)
data class CustomerDataSet(
    @JsonProperty("results")
    val customerRecords: List<CustomerMemoRecord> = emptyList()
)

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Single Record GET for one Customer Memo Record...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
data class RootJsonNodeSingle(
    @JsonProperty("d")
    val singleCustomerRecord: CustomerMemoRecord
)

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method main: Entry point for the program...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
fun main() {
    //Process User Selection...
    var answer = 0
    while (answer != 6) {
        answer = displayMainMenu()
        when (answer) {
            1 -> displayDataSet()
            2 -> displayRecord(promptTheUser("Enter a GUID to Display: "), "Current")
            3 -> createRecord()
            4 -> updateRecord()
            5 -> deleteRecord()
            6 -> {
                println("\nThank you for using this amazing program!!...Goodbye...")
                exitProcess(0)
            }
            else -> {
                println()
                println("Invalid Option Number. Please Enter a valid option number 1 thru 6.")
            }
        }
    }
}
```

```
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method promptTheUser: Prompt the end user for something...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun promptTheUser(message: String): String {
    print(message)
    return readLine()!!
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method displayMainMenu: Display the user's options and prompt for input...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun displayMainMenu(): Int {
    println()
    println("Task Options: ")
    println("(1) - Display All Data")
    println("(2) - Display a Record")
    println("(3) - Create a Record")
    println("(4) - Update a Record")
    println("(5) - Delete a Record")
    println("(6) - Exit the Program")
    print("Enter an Option Number: ")
    var num = -1
    try {
        num = readLine()!!.toInt()
    } catch (e: NumberFormatException) {
    }
    return num
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Method displaySingleRecord: Display a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
private fun displayRecord(lclGuid: String, lclTitle: String) {
    val singleRecordString = "(Guid=guid'${lclGuid}')"
    val lclUrl = MAIN_URL + singleRecordString + JSON_FORMAT
    val fetchRecord = fetchSingleRecord(lclUrl)
    showFormattedRecord( fetchRecord.singleCustomerRecord, lclTitle )
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method showFormattedRecord: Display of a single record...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun showFormattedRecord(record: CustomerMemoRecord, lclTitle: String) {
    println()
    for (x in 0..50) print("=-") // Print 50 times for line separator...
    println()
    println("$lclTitle Record for Order Number ${record.OrderNumber}:")
    for (x in 0..50) print("=-") // Print 50 times...
    println()
    println("          Guid (key): ${record.Guid}")
    println("   Date Last Updated: ${jsonDateFormatter(record.DateUpdated)}")
    println("   Time Last Updated: ${jsonTimeFormatter(record.TimeUpdated)}")
    println("Last Updated By User: ${record.LastUpdatedBy}")
    println("        Order Number: ${record.OrderNumber}")
    println("     Customer Number: ${record.CustomerNumber}")
    println("       Customer Memo: ${record.CustomerMessage}")
    for (x in 0..50) print("=-") // Print 50 times...
    println()
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method fetchSingleRecord: Fetch Data for a Single Record...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun fetchSingleRecord(lclUrl: String): FetchRecord {
    val mapper = jacksonObjectMapper()
    FuelManager.instance.baseHeaders = mapOf(
        "Authorization" to "Basic $AUTH_STRING",
        "X-CSRF-Token" to "Fetch"
    )
    //Ignore any properties that don't exist in our CustomerMemoRecord Class...
    mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
    val (_, response, result) = lclUrl.httpGet().responseString()
    val csrfToken: String? = response.headers["x-csrf-token"].elementAt(0)
    val cookie: String? = response.headers["set-cookie"].elementAt(0) + response.headers["set-
cookie"].elementAt(1)
    var memoRecord = CustomerMemoRecord()
    when (result) {
        is Result.Failure -> {
            val ex = result.getException()
            print(ex)
        }
        is Result.Success -> {
            val myJsonData: RootJsonNodeSingle = mapper.readValue(result.get())
            memoRecord = myJsonData.singleCustomerRecord
        }
    }
    return FetchRecord(
        CsrfToken = csrfToken.toString(),
        Cookie = cookie.toString(),
        singleCustomerRecord = memoRecord
    )
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method displayDataSet: Display all records in the ZTEST_KOTLIN SAP table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun displayDataSet() {
    println()
    println("...One moment please, retrieving all records...")
    FuelManager.instance.baseHeaders = mapOf("Authorization" to "Basic $AUTH_STRING")
    val mapper = jacksonObjectMapper()
    val url = MAIN_URL + JSON_FORMAT
    //This allows you to parse out only the attributes you'd like, and ignore all others...
    mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
    val (_, _, result) = url.httpGet().responseString()
    when (result) {
        is Result.Failure -> {
            val ex = result.getException()
            print(ex)
        }
        is Result.Success -> {
            val myJsonData: RootJsonNodeSet = mapper.readValue(result.get())
            val memoRecords: List<CustomerMemoRecord> = myJsonData.allRecords.customerRecords
            println("\nTable ZTEST_KOTLIN (${memoRecords.count()} Records):")
            for (x in 0..80) print("=-") // Print 80 times for line separator...
            println()
            println("Guid (key)                              " +
                    "| Date       " +
                    "| Time     " +
                    "| Updated By   " +
                    "| Order #    " +
                    "| Customer # " +
                    "| Memo")
            for (x in 0..80) print("=-") // Print 80 times...
            println()
            memoRecords.forEach {
                println(
                    "${it.Guid} " +
                            "| ${jsonDateFormatter(it.DateUpdated)} " +
                            "| ${jsonTimeFormatter(it.TimeUpdated)} " +
                            "| ${it.LastUpdatedBy.padStart(12)} " + //pad to 12 characters, to
line up with column header
                            "| ${it.OrderNumber.padStart(10)} " + //pad to 10 characters
                            "| ${it.CustomerNumber.padStart(10)} " + //pad to 10 characters
                            "| ${it.CustomerMessage}"
                )
            }
            for (x in 0..80) print("=-") // Print 80 times...
            println()
        }
    }
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method createRecord: Create a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun createRecord() {
    val lclOrder = promptTheUser("Enter Order Number: ")
    val lclCustomer = promptTheUser("Enter Customer Number: ")
    val lclMemo = promptTheUser("Enter Customer Memo: ")

    //First, fetch the CSRF Token and Cookie, prior to performing the POST...
    var url = MAIN_URL + JSON_FORMAT
    val fetchRecord = fetchSingleRecord(url)
    fetchRecord.singleCustomerRecord.OrderNumber = lclOrder
    fetchRecord.singleCustomerRecord.CustomerNumber = lclCustomer
    fetchRecord.singleCustomerRecord.CustomerMessage = lclMemo
    //Even though we are doing a POST (Create), we still need to fill in all of the
    //attributes, so enter dummy data for these ignored fields...
    fetchRecord.singleCustomerRecord.Guid = "00000000-0000-0000-0000-000000000000"
    fetchRecord.singleCustomerRecord.DateUpdated = """/Date(1578441600000)/"""
    fetchRecord.singleCustomerRecord.LastUpdatedBy = ""
    fetchRecord.singleCustomerRecord.TimeUpdated = "PT13H12M09S"
    val mapper = jacksonObjectMapper()
    // The default mapper, adjusts the field names to lower case camel case, but our
    // Gateway service has upper case (i.e. dateUpdated vs. DateUpdated),
    // so we set the UPPER_CAMEL_CASE property here...
    mapper.propertyNamingStrategy = PropertyNamingStrategy.UPPER_CAMEL_CASE
    mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false)
    // Serialize the fetchRecord Object back into a JSON String and use for our POST
    // to create the Customer Memo...
    val jsonString = mapper.writeValueAsString(fetchRecord.singleCustomerRecord)
    //Remove the "jsonFormat" URI Option, prior to doing the POST...
    url = MAIN_URL
    val newRecord: CustomerMemoRecord

    initializeFuel(fetchRecord.CsrfToken, fetchRecord.Cookie)
    val postString = """{ "d" : $jsonString }"""

    //This is a synchronous "Blocking Mode" call (i.e. will wait for a response)...
    val (_, _, result) = url.httpPost().body(postString).responseString()
    when (result) {
        is Result.Failure -> {
            println()
            println("Post Failed...")
            println(result.getException().toString() + result.error.response.toString())
        }
        is Result.Success -> {
            val myJsonData: RootJsonNodeSingle = mapper.readValue(result.get())
            newRecord = myJsonData.singleCustomerRecord
            println()
            println("...Customer Memo successfully created...")
            displayRecord(newRecord.Guid, "New")
        }
    }
```

```
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method updateRecord: Update a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun updateRecord() {
    val lclGuid = promptTheUser("Enter a GUID to Update: ")
    val newMemo = promptTheUser("Enter the New Memo: ")
    var url = "$MAIN_URL(Guid=guid'$lclGuid')$JSON_FORMAT"
    val fetchRecord = fetchSingleRecord(url)
    val originalMemo = fetchRecord.singleCustomerRecord.CustomerMessage
    fetchRecord.singleCustomerRecord.CustomerMessage = newMemo
    val mapper = jacksonObjectMapper()
    mapper.propertyNamingStrategy = PropertyNamingStrategy.UPPER_CAMEL_CASE
    val jsonString = mapper.writeValueAsString(fetchRecord.singleCustomerRecord)
    url = "$MAIN_URL(Guid=guid'$lclGuid')"

    initializeFuel(fetchRecord.CsrfToken, fetchRecord.Cookie)
    val putString = """{ "d" : $jsonString }"""
    val (_, _, result) = url.httpPut().body(putString).responseString()
    when (result) {
        is Result.Failure -> {
            println(result.getException().toString())
        }
        is Result.Success -> {
            println("...Customer Memo successfully updated...")
            println("Old Memo: $originalMemo")
            println("New Memo: $newMemo")
            displayRecord(lclGuid, "Updated")
        }
    }
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method deleteRecord: Delete a single record in the ZTEST_KOTLIN table...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun deleteRecord() {
    val lclGuid = promptTheUser("Enter a GUID to Delete: ")
    var url = "$MAIN_URL(Guid=guid'$lclGuid')$JSON_FORMAT"
    val fetchRecord = fetchSingleRecord(url)
    val mapper = jacksonObjectMapper()
    mapper.propertyNamingStrategy = PropertyNamingStrategy.UPPER_CAMEL_CASE
    url = "$MAIN_URL(Guid=guid'$lclGuid')"
    initializeFuel(fetchRecord.CsrfToken, fetchRecord.Cookie)
    val (_, _, result) = url.httpDelete().responseString()
    when (result) {
        is Result.Failure -> {
            println(result.getException().toString())
        }
        is Result.Success -> {
            println("...Customer Memo successfully deleted...")
            showFormattedRecord( fetchRecord.singleCustomerRecord, "Deleted" )
        }
    }
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
// Method initializeFuel: Set the FuelManager for UPDATE or POST...
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
private fun initializeFuel(CsrfToken: String, Cookie: String) {
    FuelManager.instance.baseHeaders = mapOf(
        "Authorization" to "Basic $AUTH_STRING",
        "X-CSRF-Token" to CsrfToken,
        "cookie" to Cookie,
        "Content-Type" to "application/json",
        "Accept" to "application/json"
    )
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Method jsonDateFormatter: Parse a JSON Date (Epoch Date) into a Java/Kotlin Date
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
private fun jsonDateFormatter(jsonDate: String): String {
    val epochDate = jsonDate.replace("[^0-9]".toRegex(), "")
    val updateDate = Instant.ofEpochMilli(java.lang.Long.parseLong(epochDate))
        .atZone(ZoneId.of("CET"))
        .toLocalDate()
    return updateDate.toString()
}

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// Method jsonTimeFormatter: Parse a JSON Time (XSD:Duration Date Type) into Java/Kotlin Time
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
private fun jsonTimeFormatter(jsonTime: String): String {
    val fTime = LocalTime.ofNanoOfDay(Duration.parse(jsonTime).toNanos())
    val df = DateTimeFormatter.ISO_LOCAL_TIME
    return fTime.format(df)
}
```

References

A simple GET Kotlin project with SAP Gateway:

https://blogs.sap.com/2018/05/21/kotlin-sap-gateway-and-intellij/
Maintain your ZTEST_KOTLIN table in Excel:
https://blogs.sap.com/2019/11/13/excel-xlsx-table-maintenance/

Kotlin and Android on the SAP Cloud Platform:
https://blogs.sap.com/2019/05/24/create-an-android-sap-cloud-platform-app-using-kotlin-and-sap-fiori-mentor/

IntelliJ IDE:
https://www.jetbrains.com/idea/

To read more about the Kotlin Language:
https://kotlinlang.org/

Kotlin Fuel:
Fuel library for Kotlin HTTP requests.
https://ryanharrison.co.uk/2018/06/15/make-http-requests-kotlin.html
https://www.baeldung.com/kotlin-fuel
https://fuel.gitbook.io/documentation/

Jackson JSON Parser:
https://www.baeldung.com/jackson-kotlin

SAP Gateway Date and Time Format:
Good Blog on OData date format:
https://blogs.sap.com/2017/01/05/date-and-time-in-sap-gateway-foundation/

Time for OData 2.0 is of type "XSD:Duration Date Type"
https://www.w3.org/TR/xmlschema11-2/#dayTimeDuration

Day/Time Duration:
https://docs.oracle.com/javase/10/docs/api/javax/xml/datatype/DatatypeFactory.html

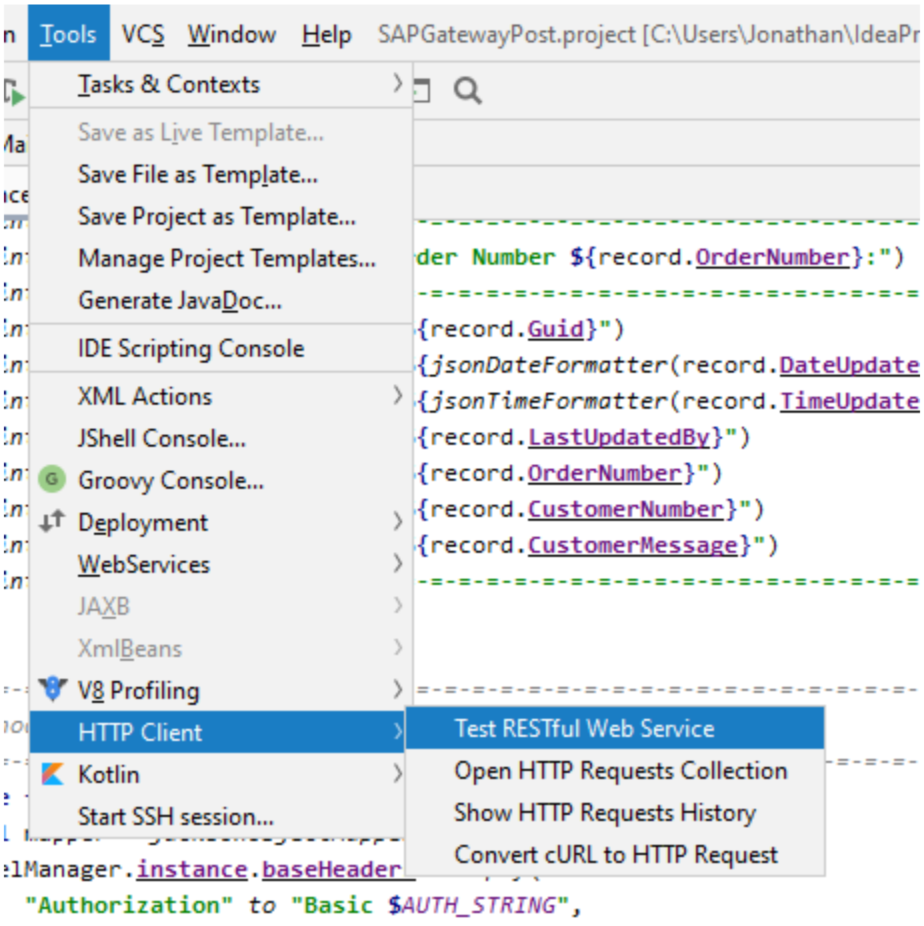
Json Date, search for code with "epochDate = jsonDate.replace". Here is an Epoch Converter:
https://www.epochconverter.com/

To Check an SAP System's Time zone, run the program TZONECHECK.

Using the IntelliJ IDE to get a Basic Authorization String:

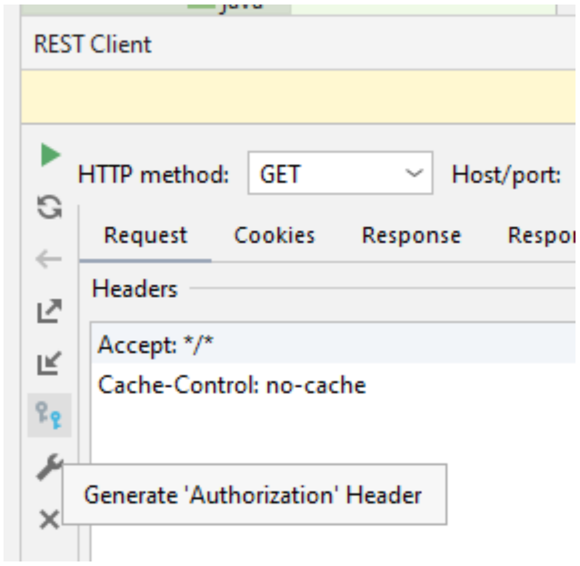
The IntelliJ IDE also has RESTful Web Service client, which allows us to generate a Basic Authorization Header string. Here are the steps:

Go to the menu path “Tools > HTTP Client > Test RESTful Web Service”:

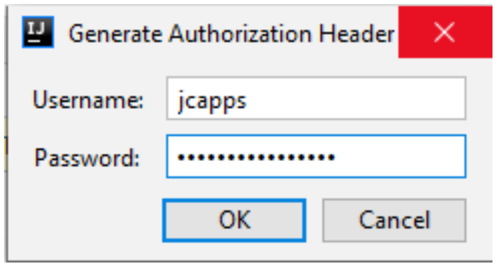


```
der Number ${record.OrderNumber}:")
{record.Guid}")
{jsonDateFormatter(record.DateUpdate
{jsonTimeFormatter(record.TimeUpdate
{record.LastUpdatedBy}")
{record.OrderNumber}")
{record.CustomerNumber}")
{record.CustomerMessage}")
lManager.instance.baseHeader
 "Authorization" to "Basic $AUTH_STRING",
```

In the lower-left, click on the button “Generate Authorization Header”:



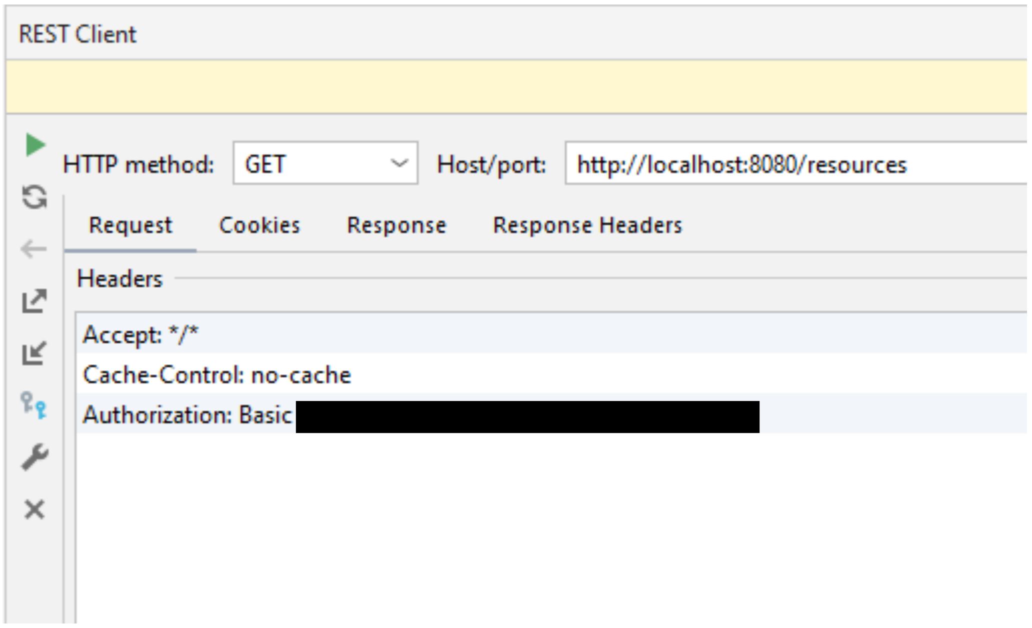
When prompted, type in your SAP User Name and password and click "OK"



Username: jcapps

Password: ••••••••••••••••

This generates an authorization string in the lower-left corner:

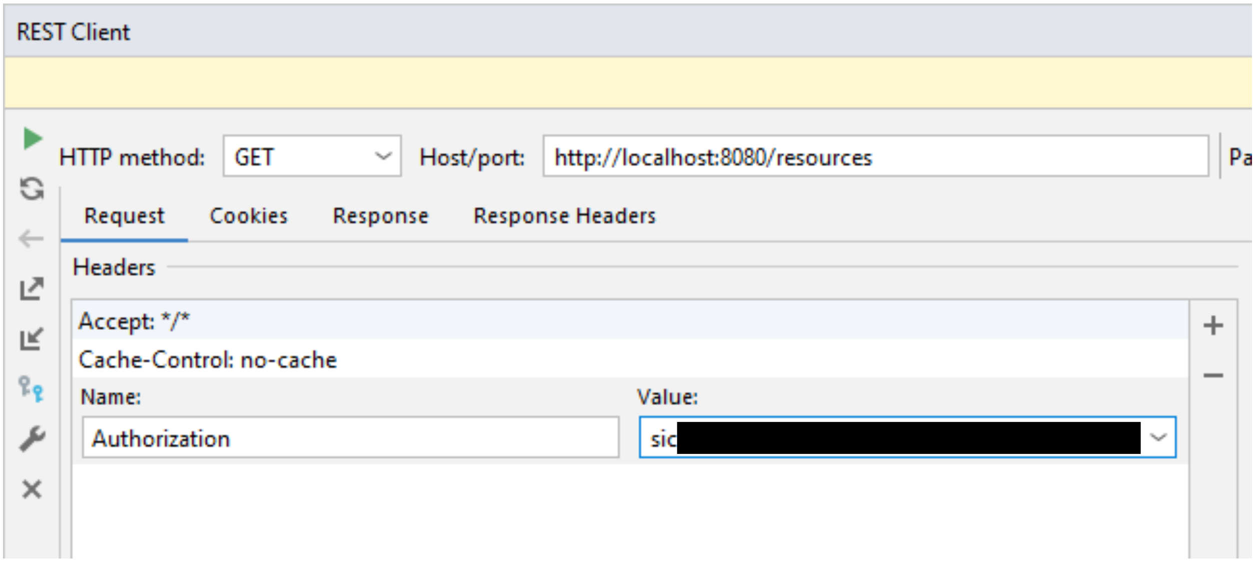


HTTP method: GET

Host/port: http://localhost:8080/resources

Accept: */*
Cache-Control: no-cache
Authorization: Basic [redacted]

Copy this string, starting with the "Basic…" part, all the way to the end of the string. When you click on the string, it will toggle into an edit box, where you may copy it:



HTTP method: GET

Host/port: http://localhost:8080/resources

Accept: */*
Cache-Control: no-cache

Name:	Value:
Authorization	sic[redacted]

Providing best-in-class solutions to meet the global demand for SAP consulting, products and implementation services.

RIZING

Rizing enables every business that uses SAP solutions to achieve a truly intelligent enterprise. We do this for Human Capital Management, Enterprise Asset Management, Consumer Industries and Geospatial Solutions with leading SAP technologies and our own deep industry experience. Rizing provides services and our proprietary apps leveraging the SAP Cloud Platform to Fortune 500 and small/medium enterprises. Our mission is always to propel your organization along the transformation journey.

rizing.com