

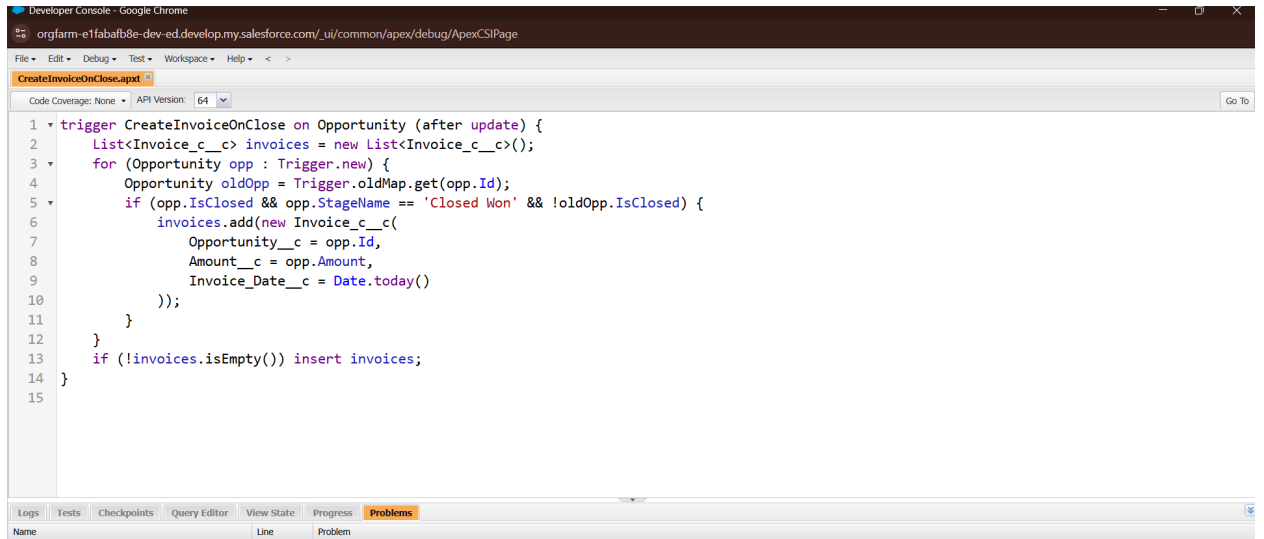
# Phase 5 – Apex Programming (Ashok Motors Capstone)

## Objective:

Develop server-side logic in Salesforce using Apex to implement complex business rules, automation, and integration that cannot be handled by declarative tools.

## 2. Apex Triggers

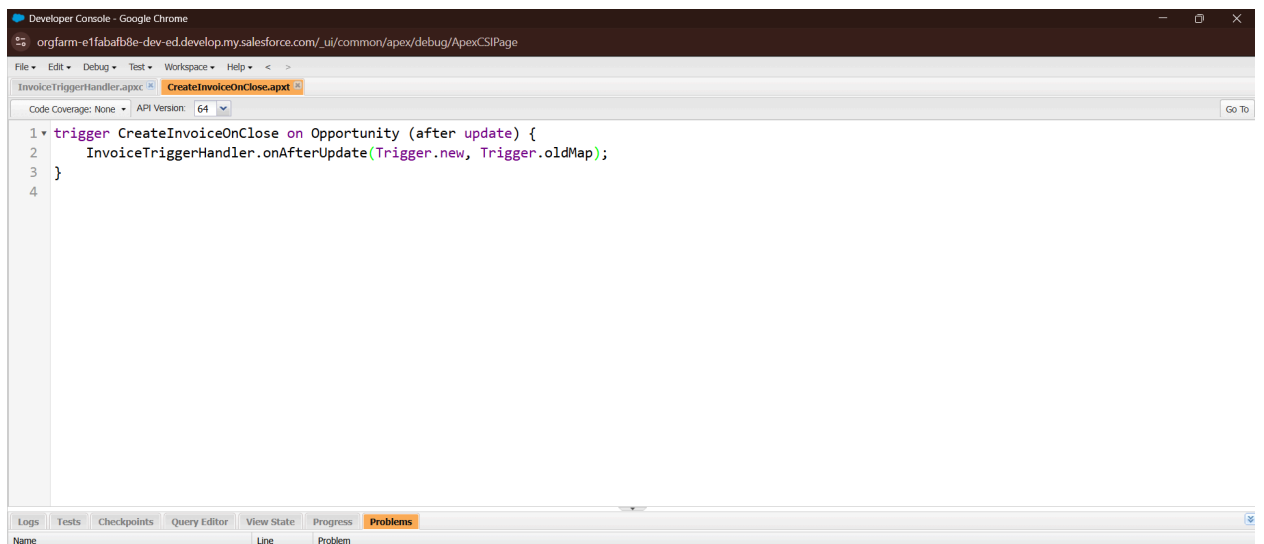
- **Purpose:** Execute logic before or after database operations (insert, update, delete).
- **Examples:**
  1. `TestDriveTrigger` (after insert) – Updates Vehicle\_\_c status to “Booked”
  2. `InvoiceTrigger` (before insert) – Validates Amount\_\_c and applies automatic discount if applicable
- **Trigger Design Pattern:** Used **one trigger per object**, handler class for logic separation
- **Screenshot:**
- **CreateInvoiceOnClose (Trigger):** Automatically creates an Invoice record when an Opportunity is marked Closed Won.
- **Test\_CreateInvoice (Class):** Verifies the trigger works correctly and ensures at least 75% code coverage for deployment



The screenshot shows the Salesforce Developer Console with the 'CreateInvoiceOnClose.apxt' trigger code. The code is as follows:

```
1 trigger CreateInvoiceOnClose on Opportunity (after update) {
2     List<Invoice_c__c> invoices = new List<Invoice_c__c>();
3     for (Opportunity opp : Trigger.new) {
4         Opportunity oldOpp = Trigger.oldMap.get(opp.Id);
5         if (opp.IsClosed && opp.StageName == 'Closed Won' && !oldOpp.IsClosed) {
6             invoices.add(new Invoice_c__c(
7                 Opportunity__c = opp.Id,
8                 Amount__c = opp.Amount,
9                 Invoice_Date__c = Date.today()
10            ));
11        }
12    }
13    if (!invoices.isEmpty()) insert invoices;
14 }
15
```

The bottom of the console shows tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected, showing a table with columns for Name, Line, and Problem.



The screenshot shows the Salesforce Developer Console with the 'InvoiceTriggerHandler.apxc' class code. The code is as follows:

```
1 trigger CreateInvoiceOnClose on Opportunity (after update) {
2     InvoiceTriggerHandler.onAfterUpdate(Trigger.new, Trigger.oldMap);
3 }
4
```

The bottom of the console shows tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected, showing a table with columns for Name, Line, and Problem.

## 9. Test Classes

- **Purpose:** Ensure at least 75% code coverage for deployment and validate logic

- **Examples:**

1. **TestVehicleManager** – Inserts sample Vehicle\_\_c records and validates status updates
2. **TestInvoiceProcessor** – Tests invoice creation and discount logic

- **Screenshot:**

The screenshot displays the Salesforce Developer Console interface. The top pane shows the source code for a test class named `Test_CreateInvoice`. The code is as follows:

```
1 @isTest
2 private class Test_CreateInvoice {
3     @isTest static void testInvoiceCreation() {
4
5         Account acc = new Account(Name='Test Acc');
6         insert acc;
7         Opportunity opp = new Opportunity(
8             Name='Test Opp',
9             StageName='Prospecting',
10            CloseDate=Date.today().addDays(10),
11            AccountId=acc.Id,
12            Amount=1000
13        );
14        insert opp;
15
16
17        Test.startTest();
18        opp.StageName = 'Closed Won';
19        update opp;
20        Test.stopTest();
21    }
22 }
```

The bottom pane shows the 'Tests' tab with a table of test run results:

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	TestRun @ 10:35:31 am			0	1
✓	Test_CreateInvoice			0	1
✓	TestRun @ 10:37:36 am			0	1

To the right of the test run table is the 'Overall Code Coverage' section, which shows the following data:

Class	Percent	Lines
Overall	100%	
CreateInvoiceOnClose	100%	1/1
InvoiceTriggerHandler	100%	10/10