WhatsNext Vision Motors
Salesforce CRM Implementation

Phase 8

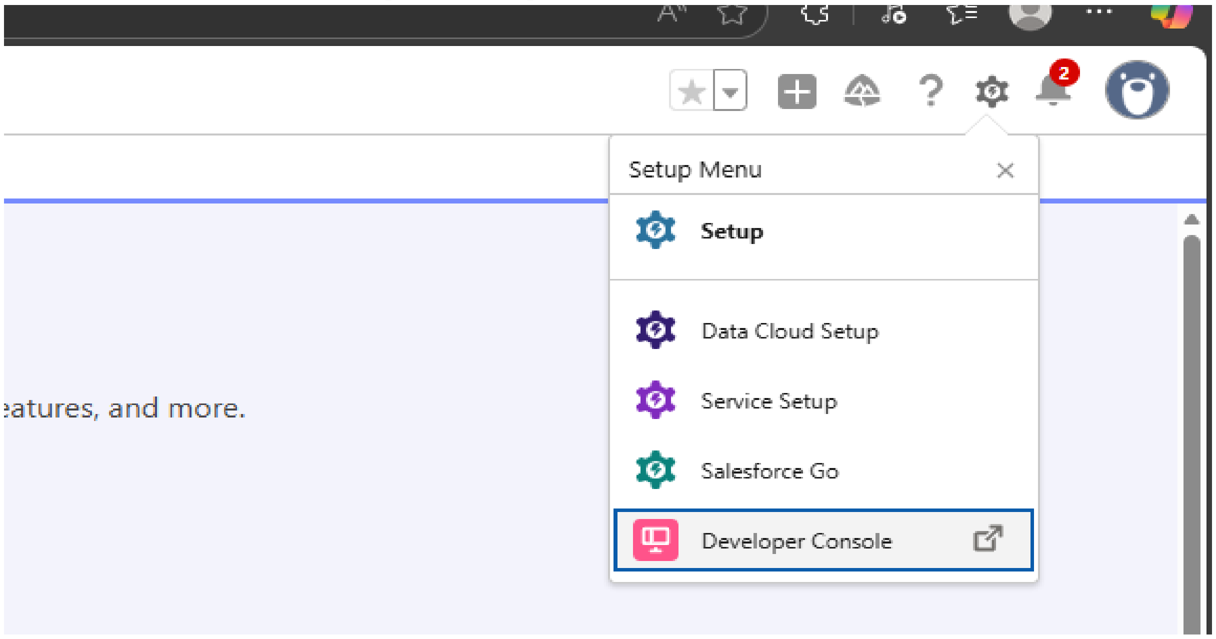
Apex Triggers, Batch Apex and Scheduled Apex

1. Phase Overview

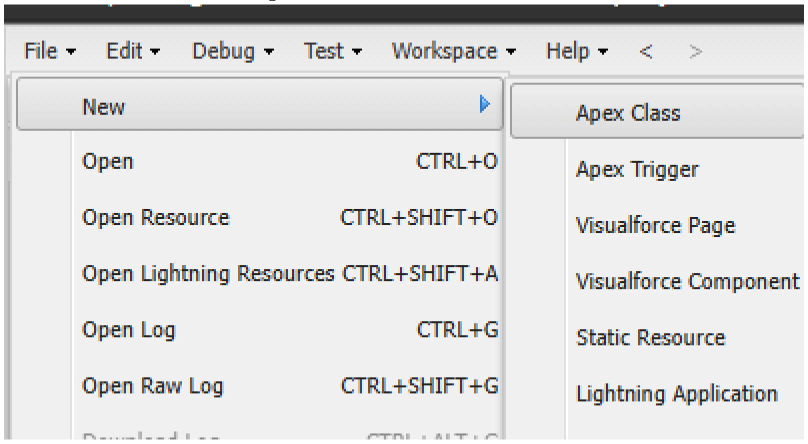
Phase 8 focuses on implementing Apex Triggers, Trigger Handler classes, Batch Apex, and Scheduled Apex to enforce business rules and automate order processing based on stock availability.

2. Creating Apex Trigger Handler Class

1. Click the Gear icon and open Developer Console



2. Click File → New → Apex Class

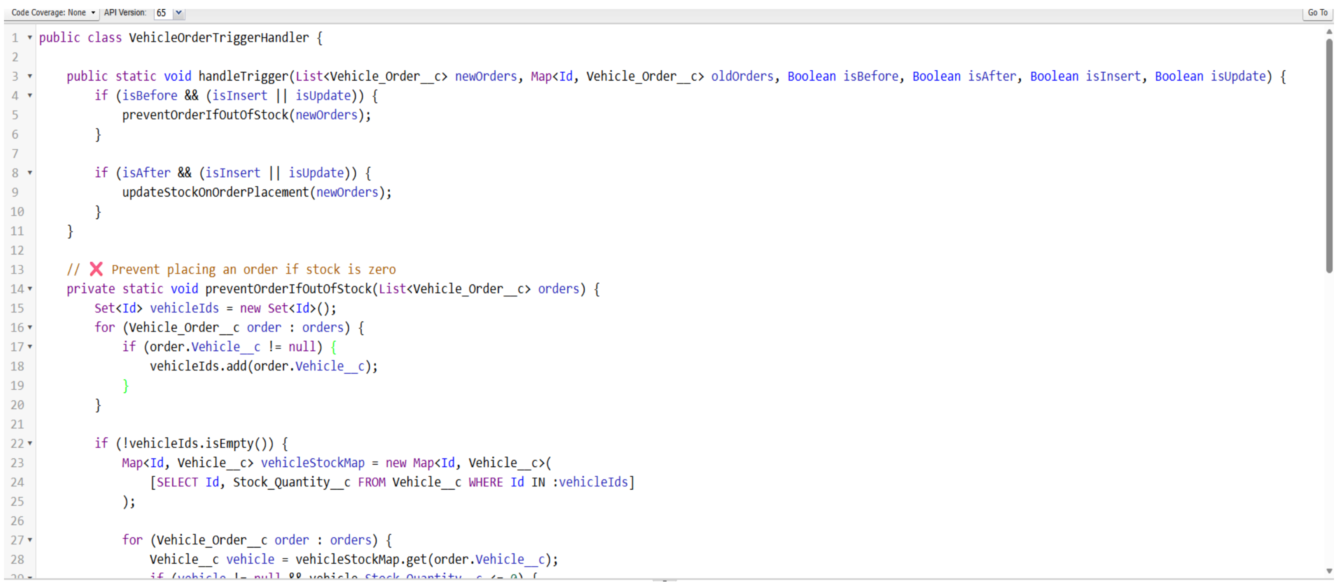


3. Enter Class Name: VehicleOrderTriggerHandler
4. Click OK and write the Apex code

Apex Class: VehicleOrderTriggerHandler

This class validates vehicle stock before order creation and updates stock quantity after order confirmation.

Code:

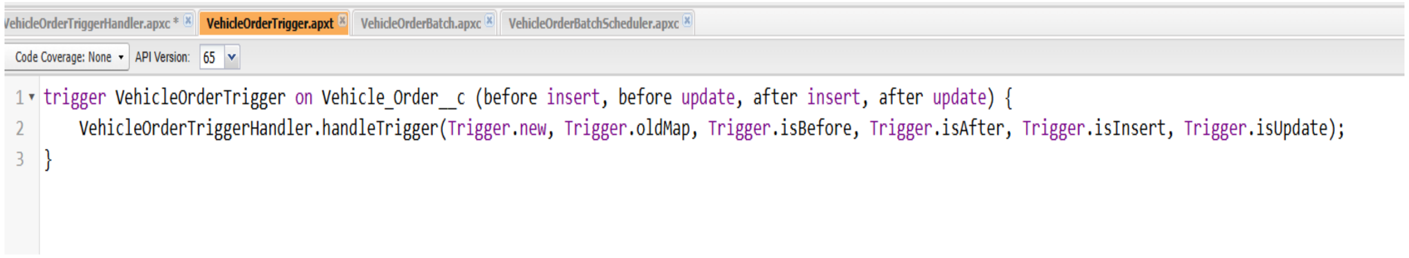


```
public class VehicleOrderTriggerHandler {

    public static void handleTrigger(List<Vehicle_Order__c> newOrders, Map<Id, Vehicle_Order__c> oldOrders, Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean isUpdate) {
        if (isBefore && (isInsert || isUpdate)) {
            preventOrderIfOutOfStock(newOrders);
        }

        if (isAfter && (isInsert || isUpdate)) {
            updateStockOnOrderPlacement(newOrders);
        }
    }

    // ❌ Prevent placing an order if stock is zero
    private static void preventOrderIfOutOfStock(List<Vehicle_Order__c> orders) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orders) {
            if (order.Vehicle__c != null) {
                vehicleIds.add(order.Vehicle__c);
            }
        }

        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            for (Vehicle_Order__c order : orders) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && vehicle.Stock_Quantity__c <= 0) {
```

3. Creating Trigger Class

- In Developer Console, click File → New → Apex Trigger
- Trigger Name: VehicleOrderTrigger
- Select Object: Vehicle_Order__c
- Call the handler class inside the trigger

Code:



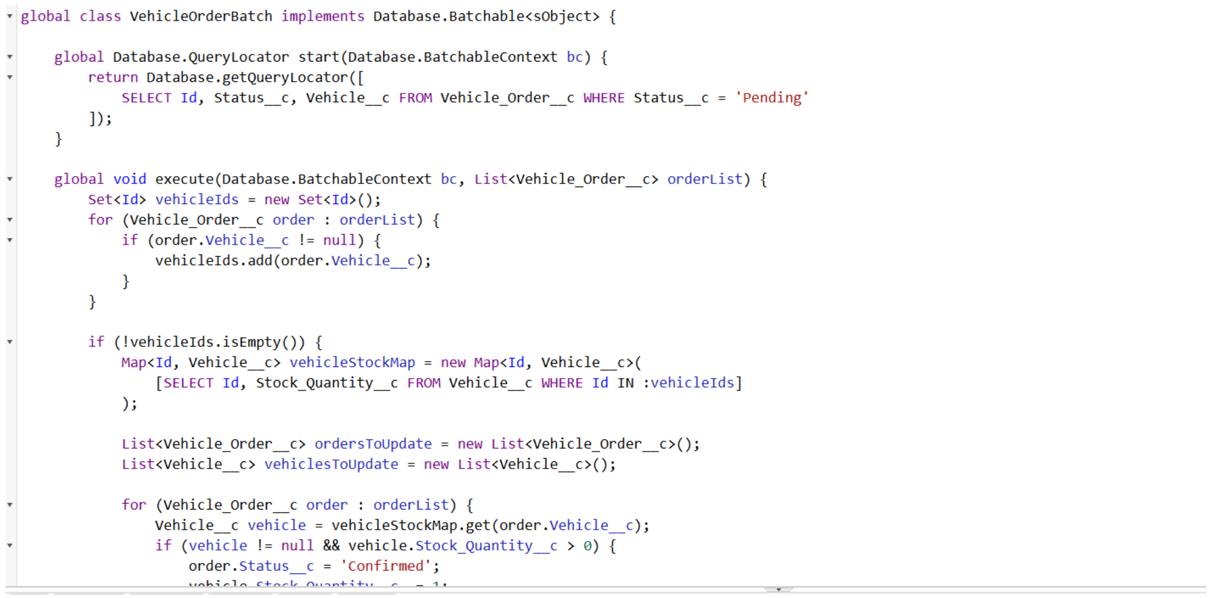
```
trigger VehicleOrderTrigger on Vehicle_Order__c (before insert, before update, after insert, after update) {
    VehicleOrderTriggerHandler.handleTrigger(Trigger.new, Trigger.oldMap, Trigger.isBefore, Trigger.isAfter, Trigger.isInsert, Trigger.isUpdate);
}
```

4. Creating Batch Apex Job

The batch job checks pending orders and confirms them when vehicle stock becomes available.

- Create a new Apex Class
- Class Name: VehicleOrderBatch
- Implement Database.Batchable interface
- Query pending orders
- Update order status and vehicle stock

Code:

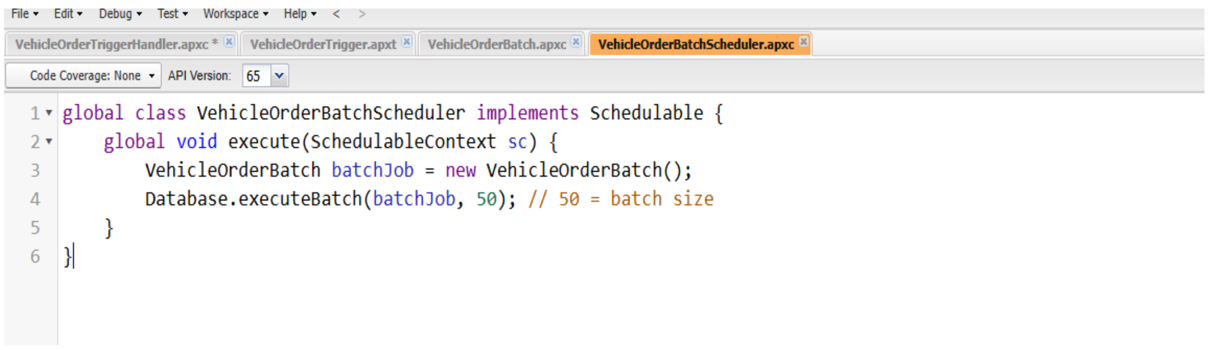


```
global class VehicleOrderBatch implements Database.Batchable<sObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Status__c, Vehicle__c FROM Vehicle_Order__c WHERE Status__c = 'Pending'
        ]);
    }

    global void execute(Database.BatchableContext bc, List<Vehicle_Order__c> orderList) {
        Set<Id> vehicleIds = new Set<Id>();
        for (Vehicle_Order__c order : orderList) {
            if (order.Vehicle__c != null) {
                vehicleIds.add(order.Vehicle__c);
            }
        }

        if (!vehicleIds.isEmpty()) {
            Map<Id, Vehicle__c> vehicleStockMap = new Map<Id, Vehicle__c>(
                [SELECT Id, Stock_Quantity__c FROM Vehicle__c WHERE Id IN :vehicleIds]
            );

            List<Vehicle_Order__c> ordersToUpdate = new List<Vehicle_Order__c>();
            List<Vehicle__c> vehiclesToUpdate = new List<Vehicle__c>();

            for (Vehicle_Order__c order : orderList) {
                Vehicle__c vehicle = vehicleStockMap.get(order.Vehicle__c);
                if (vehicle != null && vehicle.Stock_Quantity__c > 0) {
                    order.Status__c = 'Confirmed';
```

5. Creating Scheduled Apex Class

- Create a new Apex Class
- Class Name: VehicleOrderBatchScheduler
- Implement Schedulable interface
- Invoke batch class inside execute method

Code:



```
global class VehicleOrderBatchScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        VehicleOrderBatch batchJob = new VehicleOrderBatch();
        Database.executeBatch(batchJob, 50); // 50 = batch size
    }
}
```

6. Scheduling the Batch Job

The batch job is scheduled to run automatically every day to process pending vehicle orders.

7. Phase 8 Outcome

- Orders are blocked when vehicles are out of stock
- Stock is updated automatically after confirmed orders
- Pending orders are processed automatically after stock replenishment
- Nightly batch job ensures accurate order status