Phase 5: Apex Programming

Introduction

While Salesforce point-and-click automation tools (like Flows) cover most business processes, some advanced scenarios require **custom logic** that only Apex can handle. Apex (Salesforce's programming language) allows developers to create **triggers, classes, batch jobs, schedulers, and asynchronous processes** to handle complex logistics workflows, prevent booking conflicts, and integrate external systems efficiently.

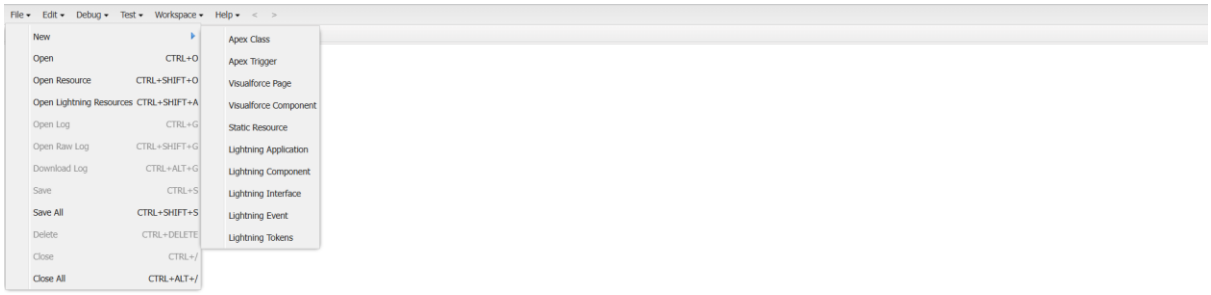
For EcoCart Logistics CRM, Apex is used to prevent overlapping shipments, handle overdue deliveries, call external courier APIs, and support bulk processing.

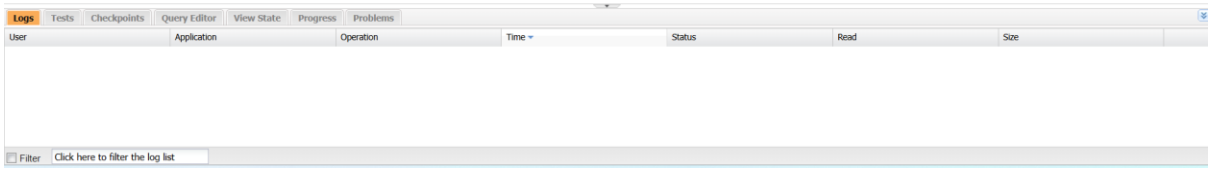
Objectives

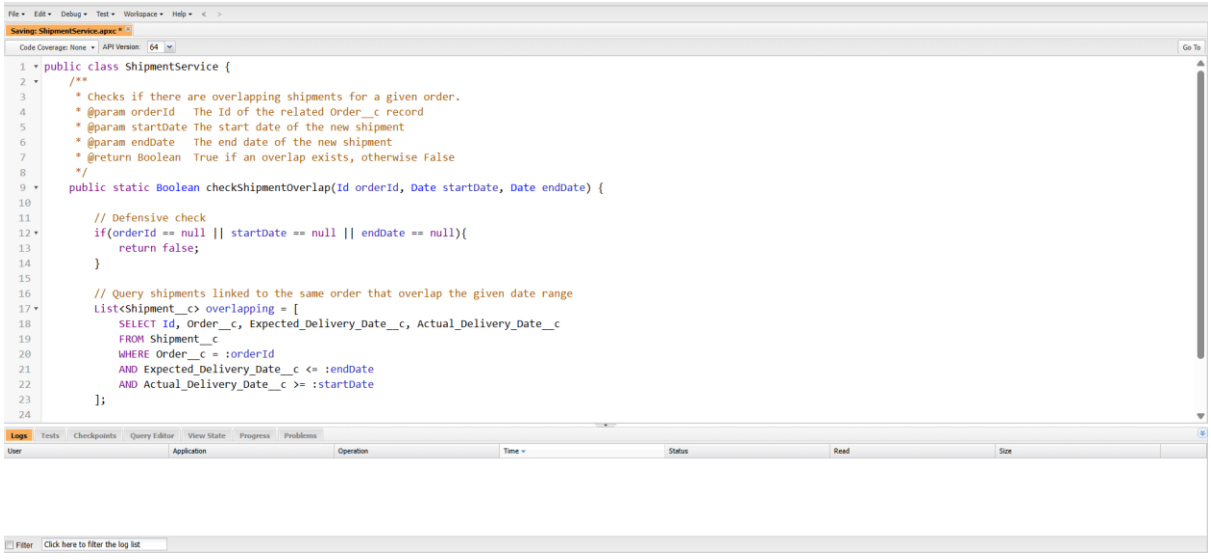
- Create reusable **Apex Classes** for shipment and order logic.
- Implement **Apex Triggers** to enforce business rules.
- Use **SOQL/SOSL** to query Orders and Shipments.
- Leverage **Collections (List, Set, Map)** for bulk processing.
- Schedule background jobs using **Batch, Queueable, and Scheduled Apex**.
- Use **Future Methods** for asynchronous integrations (e.g., courier APIs).
- Implement proper **Exception Handling** for stability.
- Write **Test Classes** to ensure code coverage and validate functionality.

1. Apex Classes & Objects

- Create a reusable **ShipmentService** class to handle logistics logic.

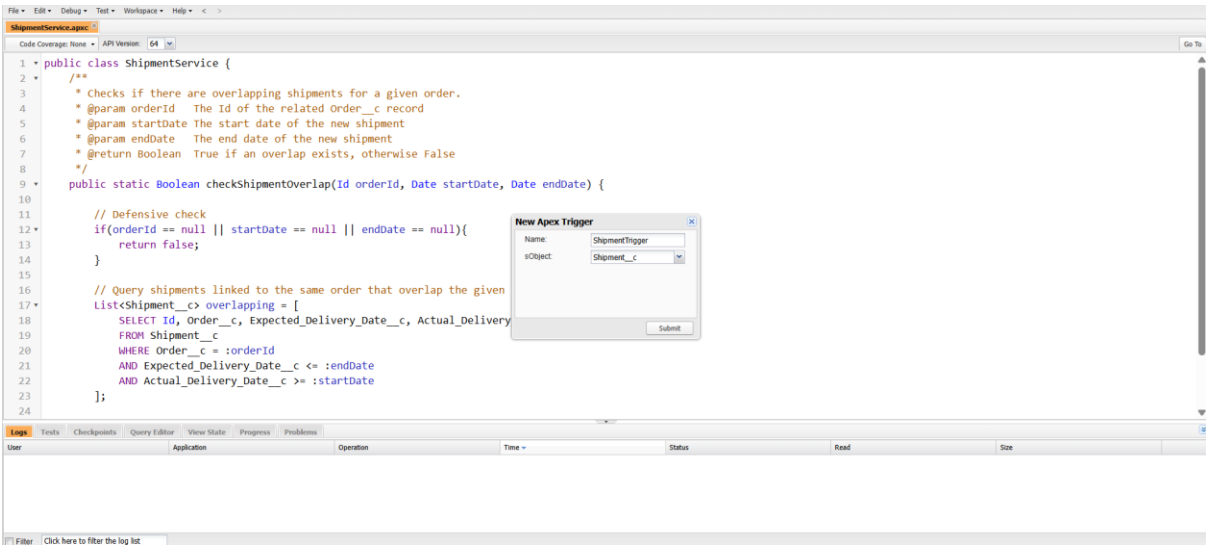






2. Trigger Using the Service

- **Trigger runs before inserting or updating a Shipment.**



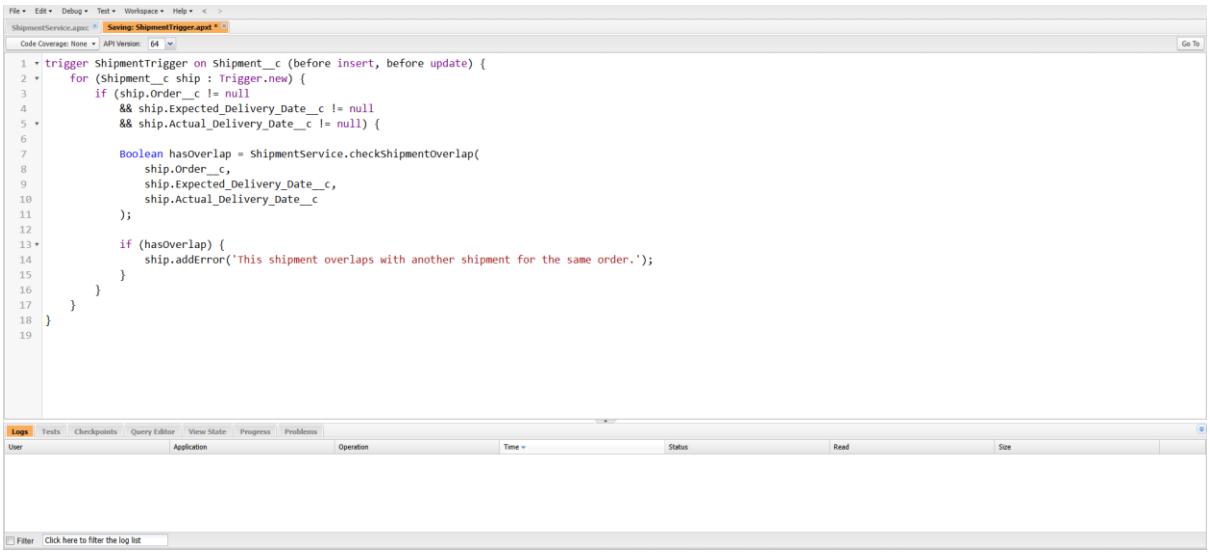
```apex
public class ShipmentService {
    /**
     * Checks if there are overlapping shipments for a given order.
     * @param orderId   The Id of the related Order__c record
     * @param startDate The start date of the new shipment
     * @param endDate   The end date of the new shipment
     * @return Boolean  True if an overlap exists, otherwise False
     */
    public static Boolean checkShipmentOverlap(Id orderId, Date startDate, Date endDate) {

        // Defensive check
        if(orderId == null || startDate == null || endDate == null){
            return false;
        }

        // Query shipments linked to the same order that overlap the given
        List<Shipment__c> overlapping = [
            SELECT Id, Order__c, Expected_Delivery_Date__c, Actual_Delivery
            FROM Shipment__c
            WHERE Order__c = :orderId
            AND Expected_Delivery_Date__c <= :endDate
            AND Actual_Delivery_Date__c >= :startDate
        ];
```

New Apex Trigger
Name: ShipmentTrigger
sObject: Shipment__c
Submit

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems
User | Application | Operation | Time | Status | Read | Size
Filter | Click here to filter the log list



```apex
trigger ShipmentTrigger on Shipment__c (before insert, before update) {
    for (Shipment__c ship : Trigger.new) {
        if (ship.Order__c != null
            && ship.Expected_Delivery_Date__c != null
            && ship.Actual_Delivery_Date__c != null) {

            Boolean hasOverlap = ShipmentService.checkShipmentOverlap(
                ship.Order__c,
                ship.Expected_Delivery_Date__c,
                ship.Actual_Delivery_Date__c
            );

            if (hasOverlap) {
                ship.addError('This shipment overlaps with another shipment for the same order.');
            }
        }
    }
}
```

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems
User | Application | Operation | Time | Status | Read | Size
Filter | Click here to filter the log list

3. SOQL & Collections

- Example: Fetch all pending shipments for an agent.

In Apex, you'll use **SOQL (Salesforce Object Query Language)** to pull data. Example (already correct in your snippet):

List<Shipment__c> pendingShipments = [SELECT Id, Name, Status__c FROM Shipment__c WHERE Status__c = 'Pending' AND Assigned_Agent__c = :UserInfo.getUserId()];

👉 This query:

- Looks in your **Shipment__c** object.
- Filters records where:
    - Status__c = 'Pending'
    - Assigned_Agent__c = Current Logged-in User

Results Achieved

- Apex Triggers enforce critical business rules (no overlapping shipments).

Next Steps

- Move to **Phase 6: User Interface Development**.
- Build **Lightning App, Record Pages, and Tabs** for Orders & Shipments.
- Develop **LWC components** (Track Shipment, Update Shipment).
- Connect LWC with Apex classes for real-time data updates.