

---

## Phase 5: Apex Programming (Developer)

### Goal

Enable advanced customization in Salesforce using **Apex programming** to handle scenarios not possible with declarative automation. This includes writing triggers, classes, queries, and test code for maintaining data integrity and automating complex telecom workflows.

---

### 1. Apex Classes & Objects

#### Implementation:

- Created CustomerHelper.cls with reusable logic to update Customer status when a Service Request is processed.

```
public class CustomerHelper {  
    public static void deactivateCustomer(Id custId) {  
        try {  
            Customer__c cust = [SELECT Id, Status__c FROM Customer__c WHERE Id = :custId LIMIT 1];  
            cust.Status__c = 'Inactive';  
            update cust;  
        } catch (Exception e) {  
            System.debug('Error while deactivating customer: ' + e.getMessage());  
        }  
    }  
}
```

**Purpose:** Centralized business logic for reusability.

The screenshot shows the Salesforce Lightning Setup console. The left sidebar contains navigation links for Setup Home, Salesforce Go, Service Setup Assistant, Commerce Setup Assistant, Field Service Setup Home (Beta), Hyperforce Assistant, Release Updates, Salesforce Mobile App, Lightning Usage, Optimizer, Sales Cloud Everywhere, ADMINISTRATION (Users, Data, Email), and PLATFORM TOOLS (Subscription Management). The main content area is titled 'Apex Classes' and shows details for the 'CustomerHelper' class. The class is active, created by 'savitri koparde' on 9/25/2025 at 8:56 PM, and has 0% code coverage. The class body is displayed in a code editor, showing the following code:

```

1 public class CustomerHelper {
2     public static void deactivateCustomers(Set<Id> custIds) {
3         if (custIds == null || custIds.isEmpty()) return;
4     }
5     List<Customer> customersToUpdate = [
6         SELECT Id, Status__c
7         FROM Customer
8         WHERE Id IN :custIds
9     ];
10
11     for (Customer c : customersToUpdate) {
12         c.Status__c = 'Inactive';
13     }
14
15     if (!customersToUpdate.isEmpty()) {
16         try {
17             update customersToUpdate;
18         } catch (DmlException e) {
19             System.debug('Error: ' + e.getMessage());
20         }
21     }
22 }
23
24 public static void deactivateCustomer(Id custId) {

```

## 2. Apex Trigger with Design Pattern

### Implementation:

- Created trigger ServiceRequestTrigger on ServiceRequest\_\_c.
- Delegated logic to ServiceRequestTriggerHandler.cls.

### Trigger:

```

trigger ServiceRequestTrigger on ServiceRequest__c (after insert, after update) {
    if (Trigger.isAfter && Trigger.isInsert) {
        ServiceRequestTriggerHandler.handleAfterInsert(Trigger.new);
    }
}

```

### Handler Class:

```

public class ServiceRequestTriggerHandler {
    public static void handleAfterInsert(List<ServiceRequest__c> newReqs) {

```

```

Set<Id> custIds = new Set<Id>();

for(ServiceRequest__c req : newReqs) {
    if(req.Request_Type__c == 'Cancellation' && req.Customer__c != null) {
        custIds.add(req.Customer__c);
    }
}

if(!custIds.isEmpty()) {
    for(Id cid : custIds) {
        CustomerHelper.deactivateCustomer(cid);
    }
}
}
}
}

```

**Purpose:** Updates related Customer to “Inactive” when a cancellation request is created.

**Apex Triggers**

This page allows you to view and modify all the triggers in your organization. To create a new trigger, navigate to the appropriate sObject trigger.

**Percent of Apex Used: 0.02%**  
You are currently using 1,418 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of a limit of 8,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

[Compile all triggers](#)

View: **All** [Create New View](#)

Action	Name ↑	Namespace Prefix	sObject Type	Api Version	Status	Size Without Comments	Last Modified By
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">ServiceRequestTrigger</a>		<a href="#">Service_Request</a>	64.0	Active	72	<a href="#">savitri koparde</a> 9/25/2025, 10:05 PM
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">ServiceRequestTriggerHandler</a>		<a href="#">Service_Request</a>	64.0	Active	79	<a href="#">savitri koparde</a> 9/25/2025, 10:50 PM
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">SRTrigger</a>		<a href="#">Service_Request</a>	64.0	Active	60	<a href="#">savitri koparde</a> 9/25/2025, 10:38 PM

### 3. SOQL Query

- Used in CustomerHelper.cls to fetch Customer details:

```
Customer__c cust = [SELECT Id, Status__c FROM Customer__c WHERE Id = :custId LIMIT 1];
```

**Purpose:** Retrieve customer data from Salesforce for processing.

---

### 4. Collections (List, Set, Map)

- **Set:** custIds ensures only unique Customers are updated.
- **List:** Trigger passes list of new ServiceRequests.
- **Map:** Could be used to map Customer Id → Customer record for bulk operations.

**Example:**

```
Map<Id, Customer__c> custMap = new Map<Id, Customer__c>(  
    [SELECT Id, Status__c FROM Customer__c WHERE Id IN :custIds]  
);
```

---

### 5. Control Statements

- **If condition** checks request type.
- **For loop** iterates over trigger records.

**Example:**

```
for(ServiceRequest__c req : newReqs) {  
    if(req.Request_Type__c == 'Cancellation') {  
        // logic  
    }  
}
```

---

### 6. Exception Handling

- Used try-catch in CustomerHelper.cls:

```
try {  
    update cust;  
} catch (Exception e) {  
    System.debug('Error: ' + e.getMessage());  
}
```

**Purpose:** Ensures errors don't break transaction flow.

---

## 7. Test Class

### Implementation:

@isTest

```
public class TestServiceRequestTrigger {
```

```
    @isTest
```

```
    static void testCancellationFlow() {
```

```
        // Create test customer
```

```
        Customer__c cust = new Customer__c(
```

```
            Name = 'Test Customer',
```

```
            Email__c = 'test@test.com',
```

```
            SIM_Number__c = '12345',
```

```
            Status__c = 'Active'
```

```
        );
```

```
        insert cust;
```

```
        // Create cancellation request
```

```
        ServiceRequest__c req = new ServiceRequest__c(
```

```
            Request_Type__c = 'Cancellation',
```

```
            Customer__c = cust.Id,
```

```
            Status__c = 'New'
```

```
        );
```

```
        insert req;
```

```
        // Fetch updated customer
```

```
        Customer__c updatedCust = [SELECT Status__c FROM Customer__c WHERE Id = :cust.Id];
```

```
        System.assertEquals('Inactive', updatedCust.Status__c);
```

```
    }
```

```
}
```

**Purpose:** Validates that trigger and helper class logic works correctly. Achieves code coverage required for deployment.

---