

# Phase 5: Apex Triggers and Classes

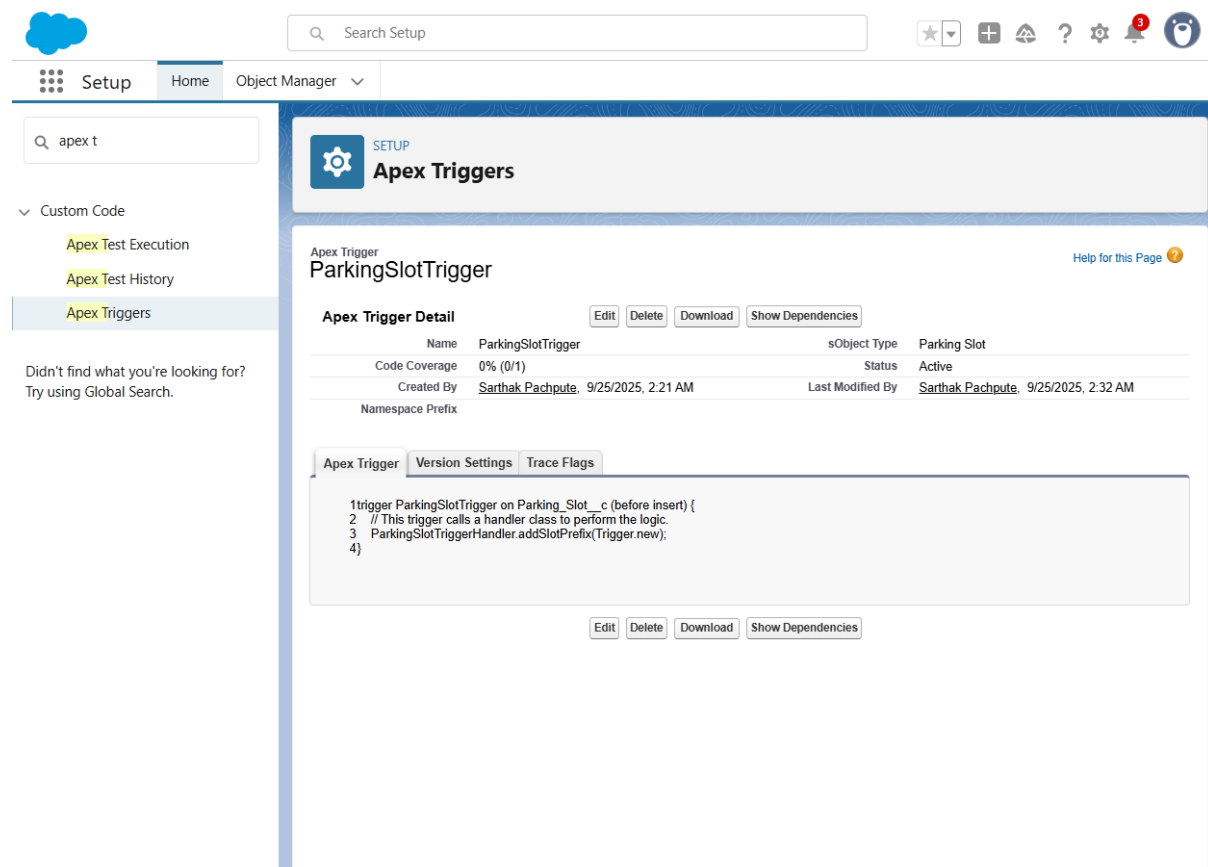
## Objective

The objective of this phase was to implement complex, server-side business logic using Apex code. While declarative tools like Flow are powerful, Apex is required for more advanced automation and custom processing. This phase focuses on using an Apex Trigger and a handler class to manage the real-time status of parking slots.

## Key Development Activities

**1. Apex Trigger (BookingTrigger)** An Apex trigger was created to execute custom logic in response to database events on the Booking object.

- **Function:** The trigger is configured to fire after a booking record is created, updated, or deleted.
- **Purpose:** It serves as the entry point for our automation. Instead of containing complex logic itself, it follows best practices by delegating the processing to a dedicated Apex handler class, ParkingSlotController.



The screenshot shows the Salesforce Setup interface. The left sidebar contains a search bar with 'apex t' and a list of items under 'Custom Code': 'Apex Test Execution', 'Apex Test History', and 'Apex Triggers'. The main content area is titled 'Apex Triggers' and shows details for a trigger named 'ParkingSlotTrigger'. The trigger is active and has 0% code coverage. The trigger code is displayed in a text area, showing a trigger on the 'Parking\_Slot\_\_c' object that calls the 'addSlotPrefix' method of the 'ParkingSlotTriggerHandler' class.

Search Setup

Setup Home Object Manager

apex t

Custom Code

- Apex Test Execution
- Apex Test History
- Apex Triggers

Didn't find what you're looking for? Try using Global Search.

### Apex Triggers

Apex Trigger ParkingSlotTrigger [Help for this Page](#)

**Apex Trigger Detail** [Edit](#) [Delete](#) [Download](#) [Show Dependencies](#)

Name	ParkingSlotTrigger	sObject Type	Parking Slot
Code Coverage	0% (0/1)	Status	Active
Created By	Sarthak Pachpute, 9/25/2025, 2:21 AM	Last Modified By	Sarthak Pachpute, 9/25/2025, 2:32 AM
Namespace Prefix			

[Apex Trigger](#) [Version Settings](#) [Trace Flags](#)

```
1 trigger ParkingSlotTrigger on Parking_Slot__c (before insert) {
2   // This trigger calls a handler class to perform the logic.
3   ParkingSlotTriggerHandler.addSlotPrefix(trigger.new);
4 }
```

[Edit](#) [Delete](#) [Download](#) [Show Dependencies](#)

**2. Apex Class (ParkingSlotController)** A separate Apex handler class, ParkingSlotController, was developed to contain the actual business logic.

- **Function:** This class includes methods that are called by the trigger. When a booking is confirmed or canceled, a method in this class is executed to find the corresponding Parking Slot and update its Status\_\_c field to "Occupied" or "Available".
- **Purpose:** Separating the logic into a handler class makes the code cleaner, easier to test, and reusable. It is the standard for professional Apex development.

The screenshot shows the Salesforce Setup interface. On the left, the 'Setup' menu is open, showing 'Custom Code' and 'Apex Classes'. The main content area is titled 'Apex Classes' and displays details for the 'ParkingSlotController' class. The class is in the 'ParkingSlotController' namespace, created by 'Sarthak Pachpute' on 9/25/2025 at 2:38 AM. The status is 'Active' and the code coverage is '0% (0/2)'. The 'Class Body' tab is selected, showing the following Apex code:

```
1 public with sharing class ParkingSlotController {
2     @AuraEnabled(cacheable=true)
3     public static integer getAvailableSlotCount() {
4         // This SOQL query counts the number of slots where the Status is 'Available'
5         return [SELECT count() FROM Parking_Slot__c WHERE Status__c = 'Available'];
6     }
7 }
```

At the bottom of the page, there is a JavaScript error message: `javascript:srcUp!["%2F005qL000008KQEfQAO%3Fisdtp%3Dp1%2...`

## Conclusion

Phase 5 marks a significant step into the programmatic capabilities of the Salesforce platform. By implementing an Apex trigger and the ParkingSlotController class, we have created a robust, real-time system for managing parking slot availability that is both powerful and scalable.