# Phase 5: Apex Programming (Developer)

**Apex Trigger – Update Show Available Seats**

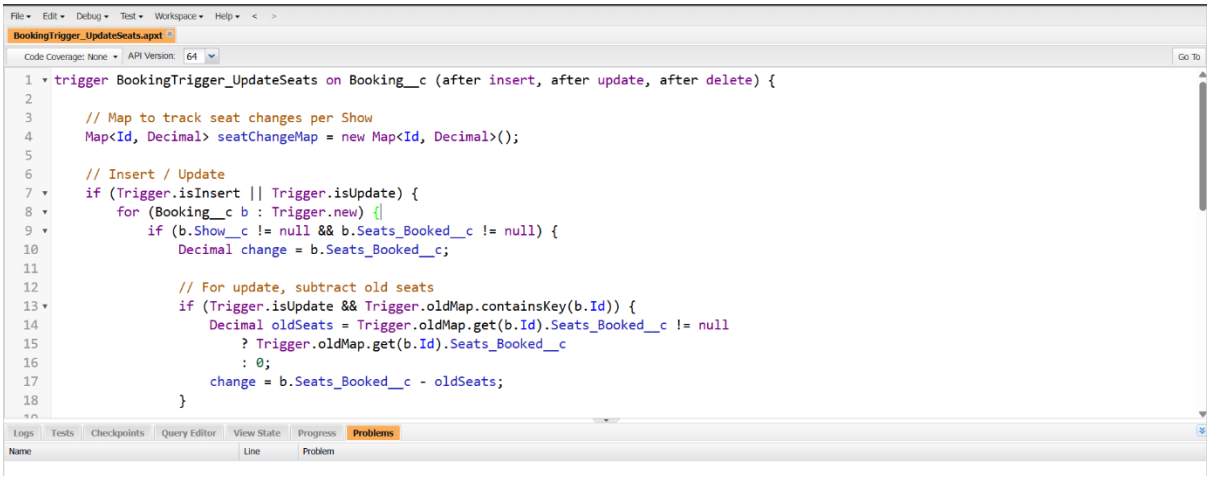
- Object: Booking__c
- Purpose: Automatically updates the available seats of a show whenever a booking is created or updated.
- When a new booking is inserted, the trigger reduces the available seats in the related Show__c record.
- When a booking is updated, it adjusts the available seats based on the change in Seats_Booked__c.
- Result: Ensures the Show__c.Available_Seats__c always reflects the correct number of seats available.

❖ **Apex Trigger Code**

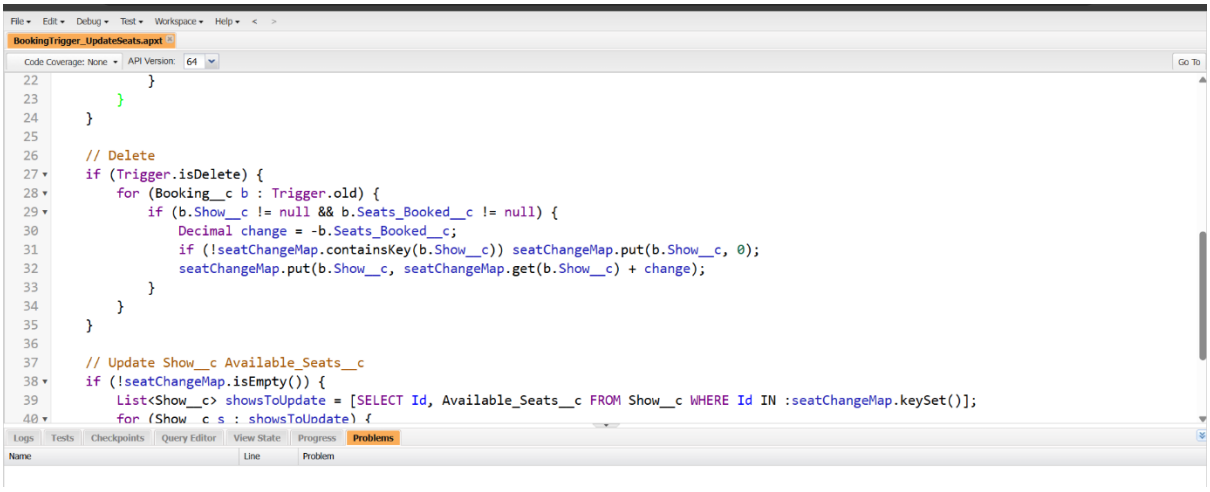
```
trigger BookingTrigger_UpdateSeats on Booking__c (after insert, after update, after delete) {
    // Map to track seat changes per Show
    Map<Id, Decimal> seatChangeMap = new Map<Id, Decimal>();
    // Insert / Update
    if (Trigger.isInsert || Trigger.isUpdate) {
        for (Booking__c b : Trigger.new) {
            if (b.Show__c != null && b.Seats_Booked__c != null) {
                Decimal change = b.Seats_Booked__c
                // For update, subtract old seats
                if (Trigger.isUpdate && Trigger.oldMap.containsKey(b.Id)) {
                    Decimal oldSeats = Trigger.oldMap.get(b.Id).Seats_Booked__c != null
                        ? Trigger.oldMap.get(b.Id).Seats_Booked__c
```

```
                    : 0;
                change = b.Seats_Booked__c - oldSeats;
            }

            if (!seatChangeMap.containsKey(b.Show__c)) seatChangeMap.put(b.Show__c,
0);
            seatChangeMap.put(b.Show__c, seatChangeMap.get(b.Show__c) + change);
        }
    }
}

// Delete
if (Trigger.isDelete) {
    for (Booking__c b : Trigger.old) {
        if (b.Show__c != null && b.Seats_Booked__c != null) {
            Decimal change = -b.Seats_Booked__c;
            if (!seatChangeMap.containsKey(b.Show__c)) seatChangeMap.put(b.Show__c,
0);
            seatChangeMap.put(b.Show__c, seatChangeMap.get(b.Show__c) + change);
        }
    }
}

// Update Show__c Available_Seats__c
if (!seatChangeMap.isEmpty()) {
```

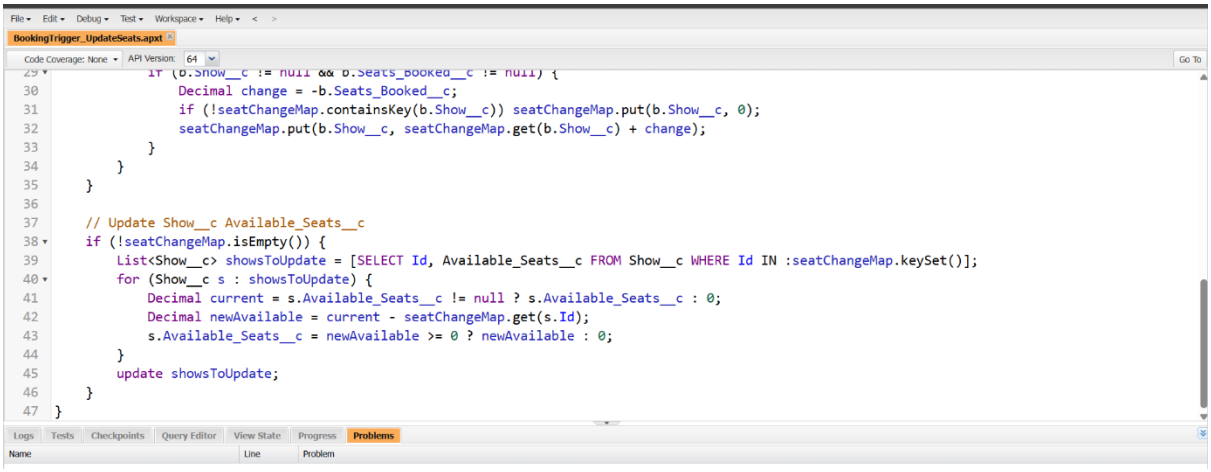
```
        List<Show__c> showsToUpdate = [SELECT Id, Available_Seats__c FROM
Show__c WHERE Id IN :seatChangeMap.keySet()];
        for (Show__c s : showsToUpdate) {
            Decimal current = s.Available_Seats__c != null ? s.Available_Seats__c : 0;
            Decimal newAvailable = current - seatChangeMap.get(s.Id);
            s.Available_Seats__c = newAvailable >= 0 ? newAvailable : 0;
        }
        update showsToUpdate;
    }
}
```



```
trigger BookingTrigger_UpdateSeats on Booking__c (after insert, after update, after delete) {

    // Map to track seat changes per Show
    Map<Id, Decimal> seatChangeMap = new Map<Id, Decimal>();

    // Insert / Update
    if (Trigger.isInsert || Trigger.isUpdate) {
        for (Booking__c b : Trigger.new) {
            if (b.Show__c != null && b.Seats_Booked__c != null) {
                Decimal change = b.Seats_Booked__c;

                // For update, subtract old seats
                if (Trigger.isUpdate && Trigger.oldMap.containsKey(b.Id)) {
                    Decimal oldSeats = Trigger.oldMap.get(b.Id).Seats_Booked__c != null
                        ? Trigger.oldMap.get(b.Id).Seats_Booked__c
                        : 0;
                    change = b.Seats_Booked__c - oldSeats;
                }
```



```
                }
            }
        }

        // Delete
        if (Trigger.isDelete) {
            for (Booking__c b : Trigger.old) {
                if (b.Show__c != null && b.Seats_Booked__c != null) {
                    Decimal change = -b.Seats_Booked__c;
                    if (!seatChangeMap.containsKey(b.Show__c)) seatChangeMap.put(b.Show__c, 0);
                    seatChangeMap.put(b.Show__c, seatChangeMap.get(b.Show__c) + change);
                }
            }
        }

        // Update Show__c Available_Seats__c
        if (!seatChangeMap.isEmpty()) {
            List<Show__c> showsToUpdate = [SELECT Id, Available_Seats__c FROM Show__c WHERE Id IN :seatChangeMap.keySet()];
            for (Show__c s : showsToUpdate) {
```

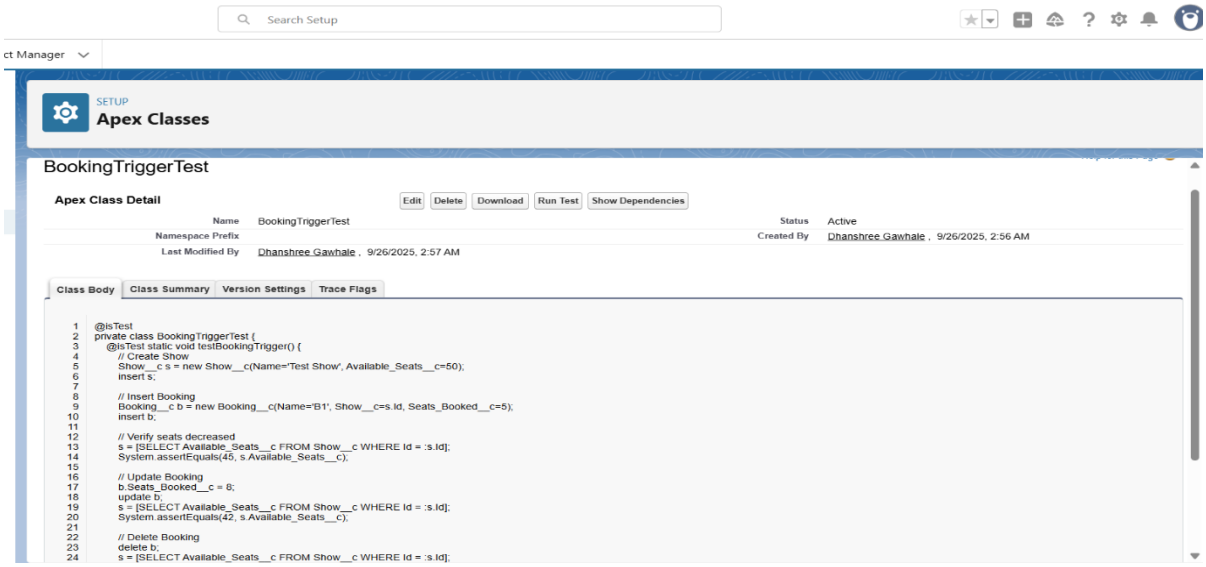


```
                if (b.Show__c != null && b.Seats_Booked__c != null) {
                    Decimal change = -b.Seats_Booked__c;
                    if (!seatChangeMap.containsKey(b.Show__c)) seatChangeMap.put(b.Show__c, 0);
                    seatChangeMap.put(b.Show__c, seatChangeMap.get(b.Show__c) + change);
                }
            }
        }

        // Update Show__c Available_Seats__c
        if (!seatChangeMap.isEmpty()) {
            List<Show__c> showsToUpdate = [SELECT Id, Available_Seats__c FROM Show__c WHERE Id IN :seatChangeMap.keySet()];
            for (Show__c s : showsToUpdate) {
                Decimal current = s.Available_Seats__c != null ? s.Available_Seats__c : 0;
                Decimal newAvailable = current - seatChangeMap.get(s.Id);
                s.Available_Seats__c = newAvailable >= 0 ? newAvailable : 0;
            }
            update showsToUpdate;
        }
    }
}
```

## ❖ Minimal Test Class for Booking Trigger

- To verify that the BookingTrigger_UpdateSeats trigger works correctly.
- When a booking is inserted, the show’s available seats decrease.
- When a booking is updated, the available seats adjust correctly.
- When a booking is deleted, the seats are restored.

```
@isTest
private class BookingTriggerTest {
    @isTest static void testBookingTrigger() {
        // Create Show
        Show__c s = new Show__c(Name='Test Show', Available_Seats__c=50);
        insert s;

        // Insert Booking
        Booking__c b = new Booking__c(Name='B1', Show__c=s.Id,
Seats_Booked__c=5);
        insert b;

        // Verify seats decreased
        s = [SELECT Available_Seats__c FROM Show__c WHERE Id = :s.Id];
        System.assertEquals(45, s.Available_Seats__c);
        // Update Booking
        b.Seats_Booked__c = 8;
        update b;
        s = [SELECT Available_Seats__c FROM Show__c WHERE Id = :s.Id];
        System.assertEquals(42, s.Available_Seats__c);
        // Delete Booking
        delete b;
        s = [SELECT Available_Seats__c FROM Show__c WHERE Id = :s.Id];
        System.assertEquals(50, s.Available_Seats__c);
    } }
```


Search Setup
ct Manager
SETUP
Apex Classes
BookingTriggerTest
Apex Class Detail
Edit
Delete
Download
Run Test
Show Dependencies
Name
BookingTriggerTest
Status
Active
Namespace Prefix
Created By
Dhanshree Gawhale , 9/26/2025, 2:56 AM
Last Modified By
Dhanshree Gawhale , 9/26/2025, 2:57 AM
Class Body
Class Summary
Version Settings
Trace Flags
@isTest
private class BookingTriggerTest {
@isTest static void testBookingTrigger() {
// Create Show
Show__c s = new Show__c(Name='Test Show', Available_Seats__c=50);
insert s;
// Insert Booking
Booking__c b = new Booking__c(Name='B1', Show__c=s.Id, Seats_Booked__c=5);
insert b;
// Verify seats decreased
s = [SELECT Available_Seats__c FROM Show__c WHERE Id = :s.Id];
System.assertEquals(45, s.Available_Seats__c);
// Update Booking
b.Seats_Booked__c = 8;
update b;
s = [SELECT Available_Seats__c FROM Show__c WHERE Id = :s.Id];
System.assertEquals(42, s.Available_Seats__c);
// Delete Booking
delete b;
s = [SELECT Available_Seats__c FROM Show__c WHERE Id = :s.Id];