

Phase 7: Integration & External Access

1. Named Credentials

Purpose: Securely store credentials for external insurance API.

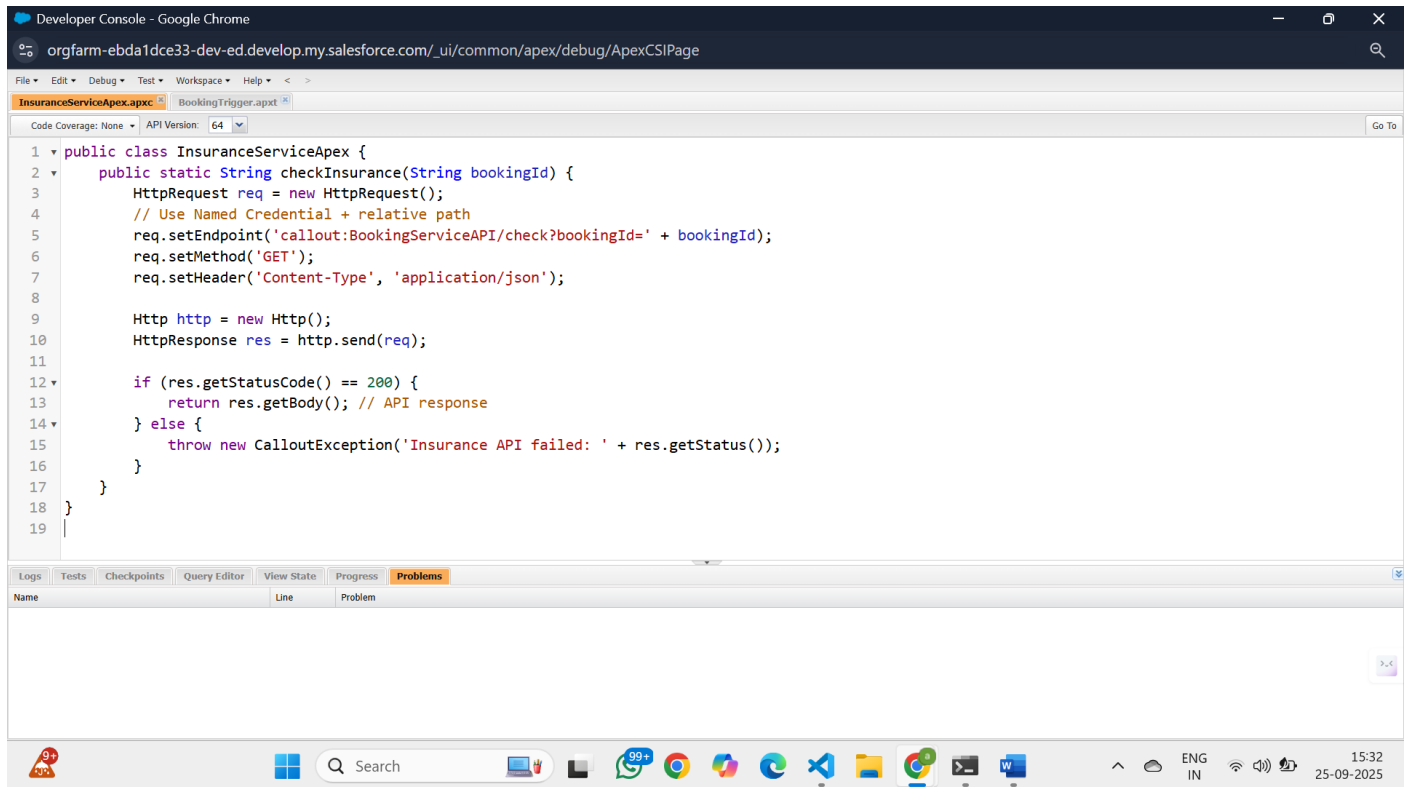
Setup:

- Go to Setup → Named Credentials → New
- Fill in:
 1. Label / Name: BookingServiceAPI
 2. URL: https://api.insurancesystem.com
 3. Identity Type: Named Principal
 4. Authentication Protocol: Password Authentication
 5. External Credential: BookingServiceCred
- Save

The screenshot displays the Salesforce Setup interface. The top navigation bar includes the Salesforce logo, a search bar, and various utility icons. The left sidebar shows the 'Setup' menu with 'Home' and 'Object Manager' options. The main content area is titled 'SETUP > NAMED CREDENTIALS' and features a header for 'BookingServiceAPI' with 'Edit' and 'Delete' buttons. The configuration form includes fields for 'Label' (BookingServiceAPI), 'Name' (BookingServiceAPI), and 'URL' (https://api.insurancesystem.com). A toggle switch for 'Enabled for Callouts' is set to 'On'. The 'Authentication' section shows 'External Credential' as 'BookingServiceCred' and 'Client Certificate' as empty. The 'Callout Options' section is also visible.

Field	Value
Label	BookingServiceAPI
Name	BookingServiceAPI
URL	https://api.insurancesystem.com
Enabled for Callouts	<input checked="" type="checkbox"/>
Authentication	
External Credential	BookingServiceCred
Client Certificate	
Callout Options	

2. Apex Class – InsuranceServiceApex

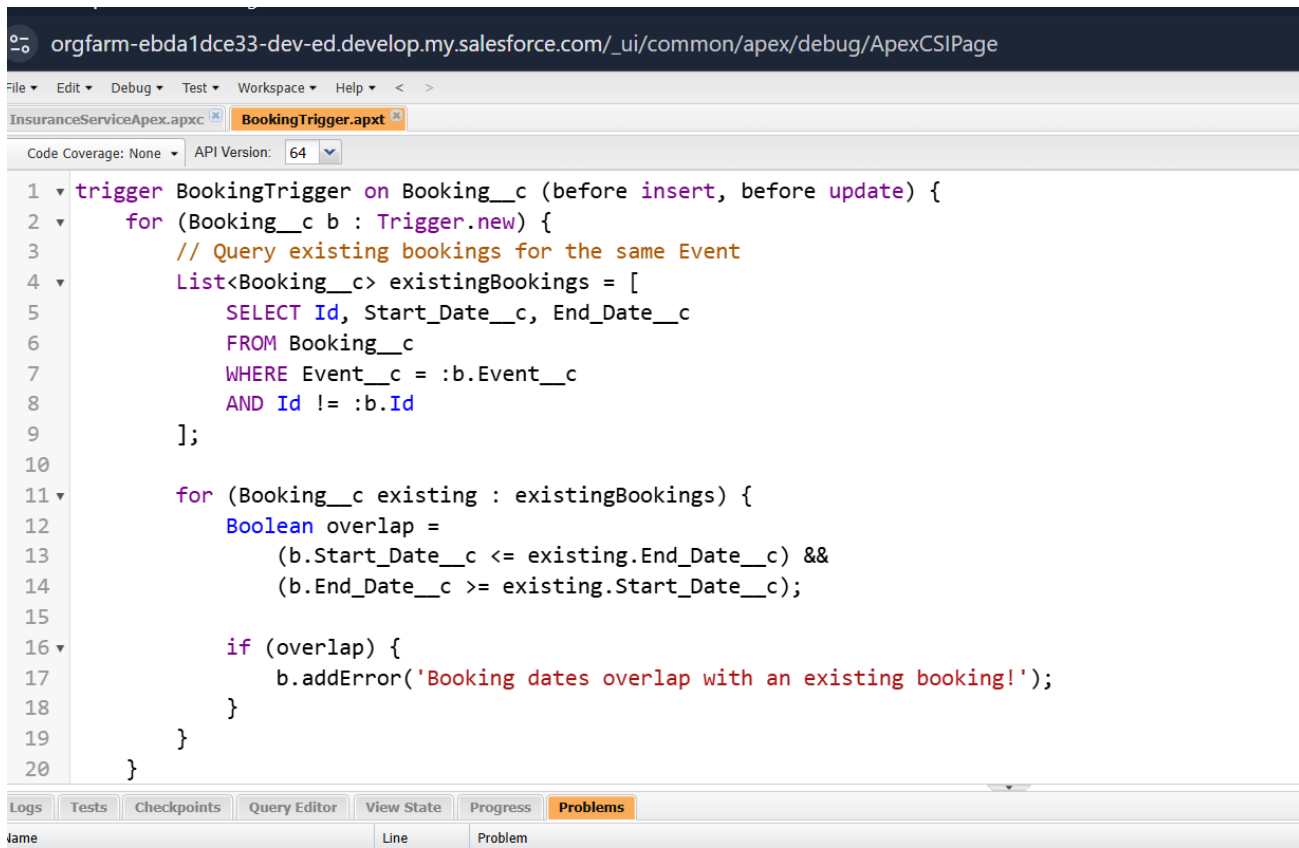


The screenshot displays the Salesforce Developer Console in Google Chrome. The browser address bar shows the URL: `orgfarm-ebda1dce33-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The console's file explorer at the top lists `InsuranceServiceApex.apxc` and `BookingTrigger.apxt`. The main editor shows the `InsuranceServiceApex` class with the following code:

```
1 public class InsuranceServiceApex {
2     public static String checkInsurance(String bookingId) {
3         HttpRequest req = new HttpRequest();
4         // Use Named Credential + relative path
5         req.setEndpoint('callout:BookingServiceAPI/check?bookingId=' + bookingId);
6         req.setMethod('GET');
7         req.setHeader('Content-Type', 'application/json');
8
9         Http http = new Http();
10        HttpResponse res = http.send(req);
11
12        if (res.getStatusCode() == 200) {
13            return res.getBody(); // API response
14        } else {
15            throw new CalloutException('Insurance API failed: ' + res.getStatus());
16        }
17    }
18 }
19
```

Below the code editor, the 'Problems' tab is active, showing a table with columns 'Name', 'Line', and 'Problem'. The table is currently empty. The Windows taskbar at the bottom shows the system clock as 15:32 on 25-09-2025.

3. Trigger – BookingInsuranceTrigger



```
1 trigger BookingTrigger on Booking__c (before insert, before update) {
2     for (Booking__c b : Trigger.new) {
3         // Query existing bookings for the same Event
4         List<Booking__c> existingBookings = [
5             SELECT Id, Start_Date__c, End_Date__c
6             FROM Booking__c
7             WHERE Event__c = :b.Event__c
8             AND Id != :b.Id
9         ];
10
11     for (Booking__c existing : existingBookings) {
12         Boolean overlap =
13             (b.Start_Date__c <= existing.End_Date__c) &&
14             (b.End_Date__c >= existing.Start_Date__c);
15
16         if (overlap) {
17             b.addError('Booking dates overlap with an existing booking!');
18         }
19     }
20 }
```

6.

Testing

1. Create a new booking in Salesforce.
2. Ensure BookingValidationTrigger prevents invalid/overlapping bookings.
3. Check Debug Logs → Insurance API Response should appear.
4. Verify Insurance_Status__c field is updated.

Phase 7 Complete:

- Secure callouts via Named Credential
- Insurance verification automated via Apex & trigger
- Booking validation ensures data integrity
- Optional advanced features (Platform Events, CDC, Connect) skipped for simplicity and project relevance