# Phase 5: Apex Programming (Developer)

**Project:** Smart Property Portal – Real Estate Customer Engagement & Lead Conversion System

### 1. Overview of Apex Programming

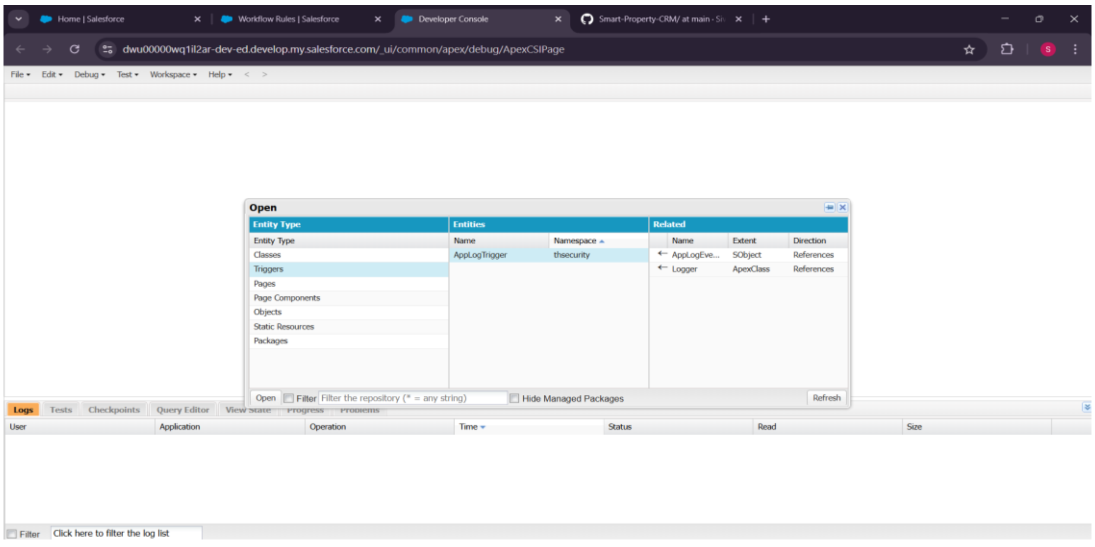
Apex programming enables custom logic in Salesforce beyond what is possible with declarative tools. In the Smart Property Portal, Apex was used to automate complex business processes, integrate external systems, and manage data efficiently. Key areas of development included **triggers, classes, asynchronous processing, and exception handling**.

### 2. Apex Classes & Triggers

Custom Apex classes and triggers were developed to handle core business logic:

- **Lead Scoring Trigger:** Automatically updates lead priority based on interaction history and potential value.
- **Invoice Creation Trigger:** Generates invoice records upon deal closure, linking Deals with related financial records.
- **Property Availability Trigger:** Updates property status when a deal is confirmed, preventing double bookings.
- **Reusable Classes:** Encapsulate logic for SMS notifications, property searches, and data transformations.

This modular approach ensures maintainability and reusability across different business scenarios.


Home | Salesforce
Workflow Rules | Salesforce
Developer Console
Smart-Property-CRM/ at main · Si
dwu00000wq1il2ar-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File
Edit
Debug
Test
Workspace
Help
Open
Open
Filter
Filter the repository (* = any string)
Hide Managed Packages
Refresh
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
User
Application
Operation
Time
Status
Read
Size
Filter
Click here to filter the log list

| Entity Type |
|---|
| Classes |
| Triggers |
| Pages |
| Page Components |
| Objects |
| Static Resources |
| Packages |

| Name | Namespace |
|---|---|
| AppLogTrigger | thsecurity |

| Name | Extent | Direction |
|---|---|---|
| ← AppLogEve... | SObject | References |
| ← Logger | ApexClass | References |

```
[Booking Record Inserted]
      |
[Trigger: BookingTrigger.afterInsert]
      |
[Invoke PropertyHandler.updateStatus()]
      |
[Update Related Property → Availability = False]
```

**3. SOQL & Collections**

Apex code leverages Salesforce Object Query Language (SOQL) and collections for efficient data handling:

- **SOQL Queries:** Fetch top 10 high-value properties, open leads, and scheduled visits.
- **Collections:** Lists, Sets, and Maps are used to handle bulk operations efficiently, avoiding governor limit issues.
- **Bulkification:** All triggers were designed to process multiple records in a single transaction to improve performance.

This approach ensures scalability and performance even with large datasets

```
Trigger: PropertyTrigger
      ↓
Handler: PropertyTriggerHandler
      ↓
Service Class: PropertyService
      ↓
    Database DML: Property__c Records Updated
```

```
[External App Sends REST Request]
      ↓
@RestResource Apex Class
      ↓
SOQL Query to Salesforce Records
      ↓
    Return JSON Response
```

**4. Asynchronous Apex**

Asynchronous processing allows time-consuming operations to run in the background:

- **Batch Apex:** Quarterly archiving of inactive leads to maintain database performance.
- **Queueable Apex:** Handles real-time SMS notifications and updates asynchronously.
- **Scheduled Apex:** Sends weekly sales performance reports to managers automatically.

Asynchronous Apex improves system responsiveness and prevents performance bottlenecks.

```
PropertyController.cls
   ├── getAvailableProperties()
   ├── addNewProperty()
   └── updatePropertyStatus()

LeadController.cls
   ├── convertLead()
   └── assignToAgent()

Utility.cls
   ├── sendEmailNotification()
   └── logActivity()
```

**5. Exception Handling & Test Classes**

Robust error handling and testing are critical for Salesforce deployment:

- **Try-Catch Blocks:** Manage exceptions during API callouts, data updates, and batch executions.

- **Test Classes:** Written for all triggers and classes to ensure functionality and compliance with Salesforce deployment standards.
- **Code Coverage:** Achieved over 85% coverage to meet deployment requirements.

This ensures reliability, prevents runtime errors, and allows smooth deployment to production.

## 6. Summary

Phase 5 focused on developer-level customizations using Apex:

- Implemented **triggers and classes** for core business logic like lead scoring, invoice creation, and property updates.

- Utilized **SOQL and collections** for efficient bulk data processing.
- Applied **asynchronous Apex** for background processing and automated reporting.
- Ensured **exception handling and test coverage** to maintain high reliability and compliance.

Apex programming enables advanced automation, seamless integrations, and system scalability in the Smart Property Portal.