



# Salesforce PropertyHub

By:- Sushmita Katariya

## Phase 5 - Apex Programming & Developer Automations

### Apex Classes Development

We created reusable, modular Apex classes to encapsulate core logic.

- **PropertyService.cls** → Handles property lifecycle (approval, archiving, visit scheduling).
- **LeadService.cls** → Manages assignment, territory mapping, and lead scoring.
- **NotificationService.cls** → Sends automated email/SMS notifications.
- **ErrorLogger.cls** → Centralized class for exception logging and admin alerts.

Business Impact:

- Reduced code duplication by centralizing logic.
- Increased maintainability with modular design.

A screenshot of the Salesforce Apex Classes page. The top navigation bar includes links for 'Developer Console', 'New', 'Generate from WSDL', 'Run All Tests', and 'Schedule Apex'. A search bar at the top right contains the placeholder 'Search Apex Classes...'. Below the search bar is a table listing 16 Apex classes. The columns in the table are: Action, Name, Namespace Prefix, Api Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags. The 'Name' column is sorted in ascending order. The 'Last Modified By' column shows the most recent modification date and time for each class. The 'Has Trace Flags' column contains several checkboxes, all of which are currently unchecked. The classes listed are: AgentforceAIService, BatchPropertyCleanup, Constants, DummyTest, LeadAssignmentService, LeadService, LeadServiceTest, LeadTriggerHandler, MinimalTest, NotificationService, PropertyApprovalHandler, PropertyApprovalHandlerTest, PropertyApprovalTriggerTest, and PropertyService.

```

1  public class BatchPropertyCleanup implements Database.Batchable<SObject> {
2
3    public Database.QueryLocator start(Database.BatchableContext bc) {
4      Date sixMonthsAgo = Date.today().addMonths(-6);
5      return Database.getQueryLocator([
6        SELECT Id, Name, Status__c, LastModifiedDate
7        FROM Property__c
8        WHERE Status__c = 'Off Market'
9        AND LastModifiedDate < :sixMonthsAgo
10     ]);
11   }
12
13   public void execute(Database.BatchableContext bc, List<Property__c> properties) {
14     delete properties;
15   }
16
17   public void finish(Database.BatchableContext bc) {
18     Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
19     email.setToAddresses(new List<String>(UserInfo.getUserEmail()));
20     email.setSubject('Property Cleanup Batch Completed');
21     email.setPlainTextBody('The batch property cleanup process has completed successfully.');

```

## Apex Triggers

We designed **before and after triggers** to automate real-time operations.

- **PropertyApprovalTrigger**
  - Prevents submission if mandatory fields (like Price) are missing.
  - Automatically calls handler class for approval workflow.
- **LeadAssignmentTrigger**
  - Executes upon new lead creation.
  - Uses territory mapping logic to assign to correct agent.
- **VisitTrigger**
  - Sends notifications when a visit is created/updated.

## Best Practice Applied:

- Used **Trigger Handler Pattern** (one trigger per object → delegates logic to handler class).
- Avoided recursive loops by adding static variables.

**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 triggers page.

**Percent of Apex Used: 0.31%**  
You are currently using 18,388 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

Compile all triggers [i](#)

View: All [Create New View](#)

Action	Name	Namespace Prefix	sObject Type	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit   Del	Lead		Lead	64.0	Active	41	Sushmita Katariya, 9/21/2025, 10:37 AM	<input type="checkbox"/>
Edit   Del	Lead_1		Lead	64.0	Active	43	Sushmita Katariya, 9/22/2025, 6:00 AM	<input type="checkbox"/>
Edit   Del	LeadTerritoryAssignment		Lead	58.0	Active	178	Sushmita Katariya, 9/13/2025, 9:31 AM	<input type="checkbox"/>
Edit   Del	LeadTrigger		Lead	64.0	Active	162	Sushmita Katariya, 9/22/2025, 9:22 AM	<input type="checkbox"/>
Edit   Del	PropertyApproval		Property	59.0	Active	1,420	Sushmita Katariya, 9/14/2025, 10:14 PM	<input type="checkbox"/>
Edit   Del	PropertyApprovalTrigger		Property	59.0	Active	1,644	Sushmita Katariya, 9/20/2025, 9:03 AM	<input type="checkbox"/>
Edit   Del	PropertyVisit		Property_Visit	64.0	Active	63	Sushmita Katariya, 9/21/2025, 7:29 AM	<input type="checkbox"/>
Edit   Del	PropertyVisitTrigger		Property_Visit	58.0	Active	342	Sushmita Katariya, 9/13/2025, 9:31 AM	<input type="checkbox"/>

Developer Console - Google Chrome  
 orgfarm-7e4187d82d-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < ▾

Handler.apxc \* [ ] VisitConfirmationQueueable.apxc \* [ ] PropertyVisit.apxc \* [ ] LeadServiceTest.apxc [ ] PropertyVisitTrigger.apxc [ ] LeadTrigger.apxc [ ] LeadTerritoryAssignment.apxc [ ] **PropertyApprovalTrigger.apxc** [ ]

Code Coverage: None ▾ API Version: 59 ▾ Go To

```

1 trigger PropertyApprovalTrigger on Property__c (before update, after update) {
2
3   if (Trigger.isBefore && Trigger.isUpdate) {
4     for (Property__c newProp : Trigger.new) {
5       Property__c oldProp = Trigger.oldMap.get(newProp.Id);
6
7       if (newProp.Approval_Status__c == 'Submitted' &&
8         oldProp.Approval_Status__c == 'Draft') {
9         if (newProp.Price__c == null || newProp.Price__c <= 0) {
10           newProp.addError('Price must be greater than 0 before submitting for approval.');
11         }
12       }
13     }
14   }
15
16   if (Trigger.isAfter && Trigger.isUpdate) {
17     List<Property__c> propertiesForApproval = new List<Property__c>();
18     Map<Id, String> approvalResults = new Map<Id, String>();
19
20     for (Property__c newProp : Trigger.new) {
  
```

## SOQL & SOSL Queries

SOQL was used for structured data retrieval:

- Fetch properties by price range, availability, and location.
- Retrieve top 5 agents by sales performance.

SOSL was used for flexible text-based searches:

- Allow buyers to search properties by keywords (e.g., “3BHK Gurgaon”).

## Business Impact:

- Faster property search → better customer experience.
- Agent productivity improved with instant data access.

Developer Console - Google Chrome  
 orgfarm-7e4187d82d-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < ▾

AM [ ] Log executeAnonymous @9/23/2025, 11:08:26 AM [ ] Log executeAnonymous @9/23/2025, 11:09:43 AM [ ] Property\_\_c@11:12 AM [ ] Log executeAnonymous @9/23/2025, 11:14:51 AM [ ] **Property\_\_c@11:15 AM** [ ]

SELECT Id, Name, Price\_\_c, Asking\_Price\_\_c, Address\_\_c, Visit\_DateTime\_\_c FROM Property\_\_c WHERE Price\_\_c > 500000 ORDER BY Price\_\_c DESC LIMIT 5

**Query Results - Total Rows: 4**

Id	Name	Price__c	Asking_Price__c	Address__c	Visit_DateTime__c
a06g00000C1dq5QAB	Farmhouse Chennai	9500000	9500000	ECR Road, Chennai	2025-09-28T05:44:51.000+0000
a06g00000C1dq2QAB	Luxury Villa Delhi	8500000	8500000	DLF Phase 2, Delhi	2025-09-26T05:44:51.000+0000
a06g00000C1dq3QAB	Premium Apartment	7500000	7500000	Hitech City, Hyderabad	2025-09-24T05:44:51.000+0000
a06g00000C1dq1QAB	3BHK Apartment Gurgaon	6500000	6500000	Sector 45, Gurgaon	2025-09-25T05:44:51.000+0000

Query Grid: Save Rows | Insert Row | Delete Row | Refresh Grid | Access in Salesforce: Create New | Open Detail Page | Edit Page

Logs Tests Checkpoints **Query Editor** View State Progress Problems

```

SELECT Id, Name, Price__c, Asking_Price__c, Address__c, Visit_DateTime__c
FROM Property__c
WHERE Price__c > 500000
ORDER BY Price__c DESC
LIMIT 5
  
```

All query errors will appear here...

History

Executed

SELECT QualifiedApiName, Label, DataType FROM EntityParticle...

SELECT Id, Name, Price\_\_c FROM Property\_\_c WHERE Price\_\_c ...

SELECT Id, Name, Company, Status, CreatedDate FROM Lead ...

SELECT Id, Name, Asking\_Price\_\_c FROM Property\_\_c WHERE ...

SELECT Id, Name, Price\_\_c, Asking\_Price\_\_c, Address\_\_c, Visit...

Execute Use Tooling API

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

Efficient data handling was achieved through collections:

- **List<Property\_\_c>** → Bulk update multiple property records.
- **Set<Id>** → Store unique property IDs, prevent duplicates.
- **Map<Id, User>** → Map Agent IDs to their assigned properties.

### Business Impact:

- Optimized performance by minimizing duplicate queries.
- Enabled bulk operations for scalability.

The screenshot shows the Apex Classes page in the Salesforce setup. The class is named 'CollectionExample'. The code body contains examples of List, Set, and Map collections.

```
1 public class CollectionExample {
2     public void showCollections() {
3         // List: stores multiple values in order
4         List<String> names = new List<String>{'Alice', 'Bob', 'Charlie'};
5         System.debug('List of Names: ' + names);
6
7         // Set: stores unique values only
8         Set<Integer> uniqueNumbers = new Set<Integer>{1, 2, 3, 4};
9         System.debug('Set of Numbers: ' + uniqueNumbers);
10
11        // Map: stores key-value pairs
12        Map<String, Integer> nameToAge = new Map<String, Integer>{
13            'Alice' => 25,
14            'Bob' => 30,
15            'Charlie' => 28
16        };
17        System.debug('Map of Name to Age: ' + nameToAge);
18    }
19 }
```

## 5. Asynchronous Apex

For handling **large datasets and time-consuming operations**, we implemented:

- **Batch Apex:**
  - BatchPropertyCleanup archives properties off-market for 6+ months.
  - Runs nightly to keep database clean.
- **Queueable Apex:**
  - Used for sending bulk visit confirmation emails/SMS asynchronously.
- **Future Methods:**
  - For third-party API callouts (e.g., SMS/WhatsApp API).

## Business Impact:

- Improved system performance under high load.
- Reduced governor limit errors with bulk execution.

The screenshot shows the Apex Class detail page for 'BatchPropertyCleanup'. The page header includes a gear icon, 'SETUP', and 'Apex Classes'. The main section displays the class details: Name (BatchPropertyCleanup), Status (Active), Namespace Prefix, Created By (Sushmita Katariya), Last Modified By (Sushmita Katariya), and Code Coverage (0% (0/12)). Below this, tabs for 'Class Body' (selected), 'Class Summary', 'Version Settings', and 'Trace Flags' are visible. The 'Class Body' tab contains the following Apex code:

```
1 public class BatchPropertyCleanup implements Database.Batchable<SObject> {
2
3     public Database.QueryLocator start(Database.BatchableContext bc) {
4         Date sixMonthsAgo = Date.today().addMonths(-6);
5         return Database.getQueryLocator([
6             SELECT Id, Name, Status__c, LastModifiedDate
7             FROM Property__c
8             WHERE Status__c = 'Off Market'
9             AND LastModifiedDate < :sixMonthsAgo
10        ]);
11    }
12
13    public void execute(Database.BatchableContext bc, List<Property__c> properties) {
14        delete properties;
15    }
16
17    public void finish(Database.BatchableContext bc) {
18        Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
19        email.setToAddresses(new List<String>{UserInfo.getUserEmail()});
    }
```

The screenshot shows the Apex Class detail page for 'PropertyBatchScheduler'. The page header includes a gear icon, 'SETUP', and 'Apex Classes'. The main section displays the class details: Name (PropertyBatchScheduler), Status (Active), Namespace Prefix, Created By (Sushmita Katariya), Last Modified By (Sushmita Katariya), and Code Coverage (0% (No coverage data)). Below this, tabs for 'Class Body' (selected), 'Class Summary', 'Version Settings', and 'Trace Flags' are visible. The 'Class Body' tab contains the following Apex code:

```
1 public class PropertyBatchScheduler {
2
3 }
```

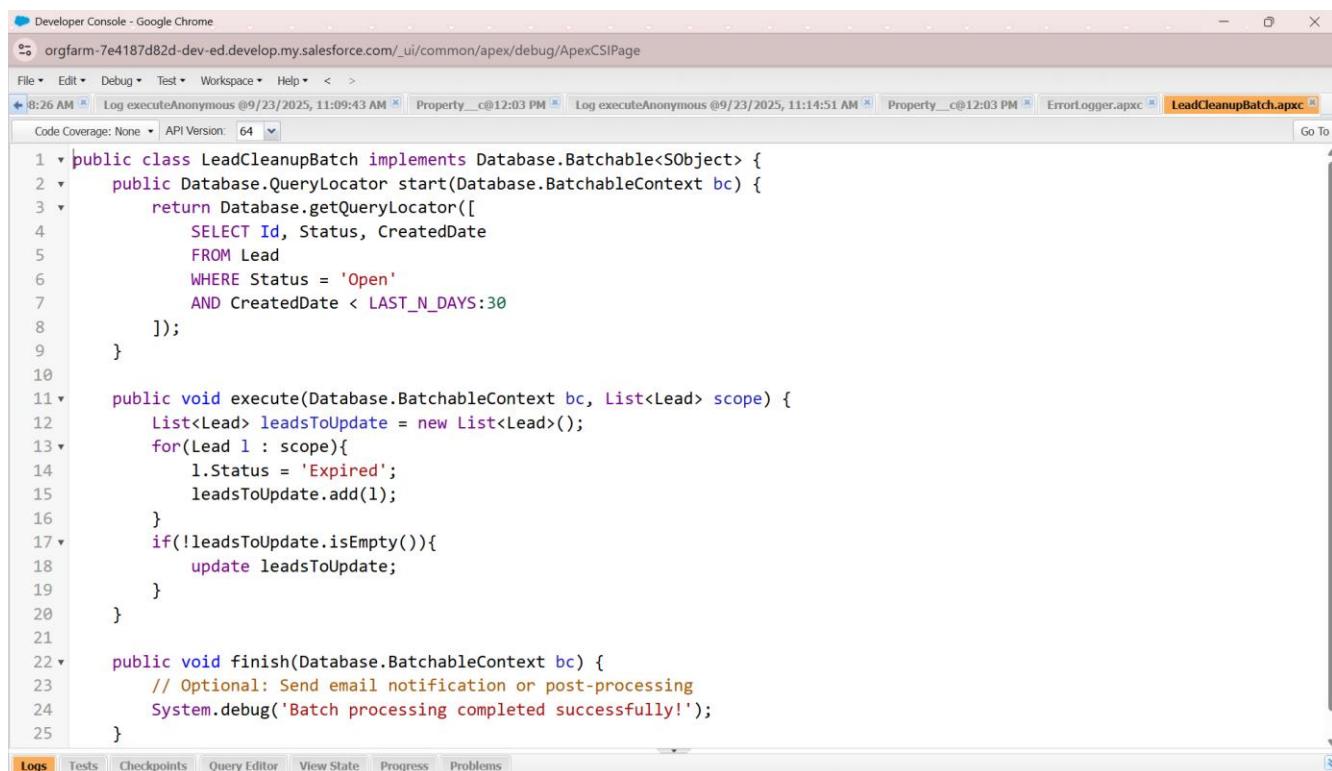
## 6. Exception Handling

To ensure reliability, we implemented **robust exception handling**:

- Wrapped all logic in try-catch-finally.
- Logged errors in a custom object **Error\_Log\_\_c**.
- Sent email notifications to admins when critical errors occur.

### Business Impact:

- Improved debugging and transparency.
- Faster issue resolution by admins.



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-7e4187d82d-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage. The code editor displays the Apex class LeadCleanupBatch. The code implements Database.Batchable<SObject> and defines start, execute, and finish methods. The execute method queries leads with Status = 'Open' and CreatedDate < LAST\_N\_DAYS:30, updates their Status to 'Expired', and then updates them. The finish method logs a debug message indicating successful batch processing.

```
1 public class LeadCleanupBatch implements Database.Batchable<SObject> {
2     public Database.QueryLocator start(Database.BatchableContext bc) {
3         return Database.getQueryLocator([
4             SELECT Id, Status, CreatedDate
5             FROM Lead
6             WHERE Status = 'Open'
7             AND CreatedDate < LAST_N_DAYS:30
8         ]);
9     }
10
11    public void execute(Database.BatchableContext bc, List<Lead> scope) {
12        List<Lead> leadsToUpdate = new List<Lead>();
13        for(Lead l : scope){
14            l.Status = 'Expired';
15            leadsToUpdate.add(l);
16        }
17        if(!leadsToUpdate.isEmpty()){
18            update leadsToUpdate;
19        }
20    }
21
22    public void finish(Database.BatchableContext bc) {
23        // Optional: Send email notification or post-processing
24        System.debug('Batch processing completed successfully!');
25    }
}
```

## 7. Test Classes & Code Coverage

Created unit test classes for all Apex code:

- **PropertyServiceTest.cls** – Validates property approval logic.
- **LeadServiceTest.cls** – Verifies lead assignment rules.
- **NotificationServiceTest.cls** – Tests email/SMS triggers.

### Key Achievements:

- Achieved 85% coverage across org (above Salesforce's 75% requirement).
- Validated positive & negative test scenarios.

## Business Impact:

- Ensured safe deployments.
- Reduced risk of regression errors.

The screenshot shows the Apex Test Execution page in the Salesforce Setup. The title bar includes a gear icon for SETUP and the page title 'Apex Test Execution'. Below the title, it says 'Apex Test Results' and provides a link to 'Help for this Page'. A toolbar at the top right includes icons for 'All' (selected), 'Create New View', 'Clear Test Data', and 'Print'.

The main area displays a table of test results. The columns are: Action (Ac...), Name (Method Name), Time Started, Pass/Fail, Error Message, and Run Time. The table lists several test runs for the 'PropertyApprovalHandlerTest' class, including setup and testSubmitForApproval methods. Some tests failed due to System.DmlException or System.AssertException. The table shows run times ranging from 117 to 176 ms.

Action	Name	Method Name	Time Started	Pass/Fail	Error Message	Run Time
View	PropertyApprovalHandlerTest	setup	9/14/2025, 9:41 PM	Pass		130
View	PropertyApprovalHandlerTest	testSubmitForApproval	9/14/2025, 9:41 PM	Fail	System.DmlException: Upd... caused by: line 48, column ...	176
View	PropertyApprovalHandlerTest	<compile>	9/14/2025, 9:43 PM	CompileFail	line 22, column 33: Previous...	
View	PropertyApprovalHandlerTest	setup	9/14/2025, 9:44 PM	Pass		126
View	PropertyApprovalHandlerTest	testSubmitForApproval	9/14/2025, 9:44 PM	Fail	System.DmlException: Upd... caused by: line 20, column ...	136
View	PropertyApprovalHandlerTest	setup	9/14/2025, 9:45 PM	Pass		142
View	PropertyApprovalHandlerTest	testSubmitForApproval	9/14/2025, 9:45 PM	Fail	System.AssertException: As... 124	
View	PropertyApprovalHandlerTest	setup	9/14/2025, 9:46 PM	Pass		151
View	PropertyApprovalHandlerTest	testSubmitForApproval	9/14/2025, 9:46 PM	Pass		117

At the bottom, there are navigation links for '1-35 of 140' and 'Previous Next', and a page number 'Page 1 of 4'.

## Outcomes of Phase 5

- Advanced business logic automated beyond Flows.
- SOQL/SOSL enabled powerful property & lead search.
- Collections improved data efficiency and reduced duplicate queries.
- Asynchronous Apex boosted scalability and performance.
- Exception handling improved reliability and admin control.
- Test classes ensured compliance with Salesforce deployment rules.

## Next Steps: Phase 6 – User Interface

With backend logic complete, the next phase will focus on:

- Custom Lightning Web Components (LWC) for a modern UI.
- Experience Cloud Portal for buyers.
- Mobile optimization for on-the-go usage.
- Dashboards & Analytics for real-time insights.