Phase 5: Apex Programming (Developer)

Apex is Salesforce's **object-oriented programming language** that allows developers to write complex business logic and automate processes that go beyond declarative tools (like Flows or Process Builder). It runs on the Salesforce platform and helps implement **customized, scalable, and efficient solutions**.

1. Classes & Objects

Purpose:

- Apex classes define reusable blocks of code that contain methods and variables.
- Objects are instances of classes used to execute logic and store temporary or permanent data.

Use Cases:

- Create a utility class to calculate recycled product cost.
- Build classes for automated notifications or custom business logic.

2. Apex Triggers (Before/After Insert, Update, Delete)

Purpose:

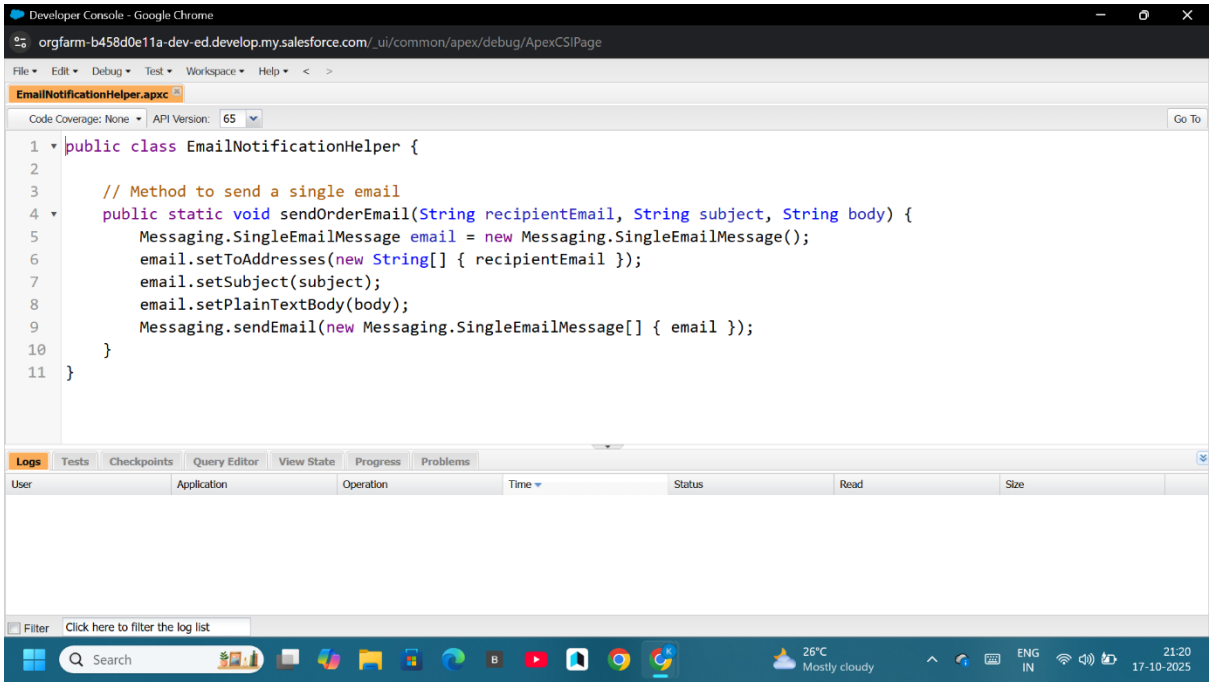
- Triggers execute Apex code automatically when records are created, updated, deleted, or undeleted.

Trigger Types:

- **Before Insert/Update:** Validate or modify records before saving.
- **After Insert/Update/Delete:** Execute logic after records are saved.

Use Cases:

- Update total waste collected after each record insertion.
- Notify recycling center when an order status changes.



3. Trigger Design Pattern

Purpose:
To maintain clean, reusable, and scalable trigger logic.

Best Practices:

- Only **one trigger per object**.
- Delegate logic to **handler classes**.
- Ensure triggers are **bulkified** (handle multiple records efficiently).

Use Case:
Maintain organized logic for Plastic Waste automation through a central handler class.

4. SOQL & SOSL

Purpose:

- **SOQL (Salesforce Object Query Language):** Fetch data from one or more Salesforce objects.
- **SOSL (Salesforce Object Search Language):** Search data across multiple objects and fields.

Use Cases:

- Retrieve all orders placed by a specific customer (SOQL).
- Search for a keyword in multiple objects (SOSL).

5. Collections: List, Set, Map

Purpose:
Collections help store and manage multiple data values efficiently.

Collection	Description	Example Use Case
List	Ordered collection of elements	List of Orders or Waste records
Set	Unordered, unique elements	Unique Waste Types
Map	Key-value pairs	Map of Customer ID → Order List

6. Control Statements

Purpose:
Used to execute code conditionally or repeatedly.

Common Statements:

- if, else, switch
- for, while, do-while
- break, continue

7. Batch Apex

Purpose:
Used for processing large datasets asynchronously in smaller batches (up to 50 million records).

Use Cases:

- Update thousands of recycled product records nightly.
- Recalculate waste totals after data migration.

8. Queueable Apex

Purpose:
Run asynchronous operations that need more complex logic than future methods.

Use Cases:

- Send confirmation emails after order approval.
- Perform real-time integrations with ERP systems.

9. Scheduled Apex

Purpose:
Run Apex code automatically at defined times or intervals.

Use Cases:

- Generate weekly reports of recycled products.
- Update order statuses every night.

10. Future Methods

Purpose:
Execute code asynchronously, especially useful for **callouts** or background processing.

Use Cases:

- Notify customers about shipment updates.
- Push order data to external ERP systems.

11. Exception Handling

Purpose:
Catch and handle errors gracefully to prevent transaction failure.

Use Cases:

- Handle record insert/update exceptions.
- Log errors for admin review.

12. Test Classes

Purpose:
Validate Apex code and ensure proper functionality before deployment.
Salesforce requires **at least 75% code coverage** for deployment to production.

Use Cases:

- Test triggers, classes, and async logic.
- Simulate user actions and validate outcomes.

13. Asynchronous Processing

Purpose:
Perform background tasks that don't block the main execution thread.
Includes **Batch Apex, Queueable Apex, Scheduled Apex, and Future Methods.**

Use Cases:

- Process large volumes of recycling data.
- Sync Salesforce with external ERP systems efficiently.
- Send notifications or update reports in the background.

✅ **Conclusion of Phase 5**

Apex Programming empowers Salesforce developers to:

- Build **custom logic** beyond declarative tools.
- Handle **large datasets efficiently**.
- Perform **automations and integrations asynchronously**.
- Ensure system stability through **exception handling and test coverage**.

By implementing Apex efficiently, the organization achieves **scalability, automation, and superior performance** in its Salesforce ecosystem.