

Phase 5: Apex Programming (Developer)

- Classes & Objects
- Apex Triggers (before/after insert/update/delete)
- Trigger Design Pattern
- SOQL & SOSL
- Collections: List, Set, Map
- Control Statements
- Batch Apex
- Queueable Apex
- Scheduled Apex
- Future Methods
- Exception Handling
- Test Classes
- Asynchronous Processing

Classes & Objects:

Objects

- Objects are database tables in Salesforce that store data.
- **Standard Objects** (like Account, Contact) come built-in.

- **Custom Objects** (like Meal_Plan__c, Recipe__c, Ingredient__c) are created to fit business needs.
- Each object can have **fields** (e.g., Quantity__c, Calories__c, Meal_Plan_Lookup__c).

Example in our project:

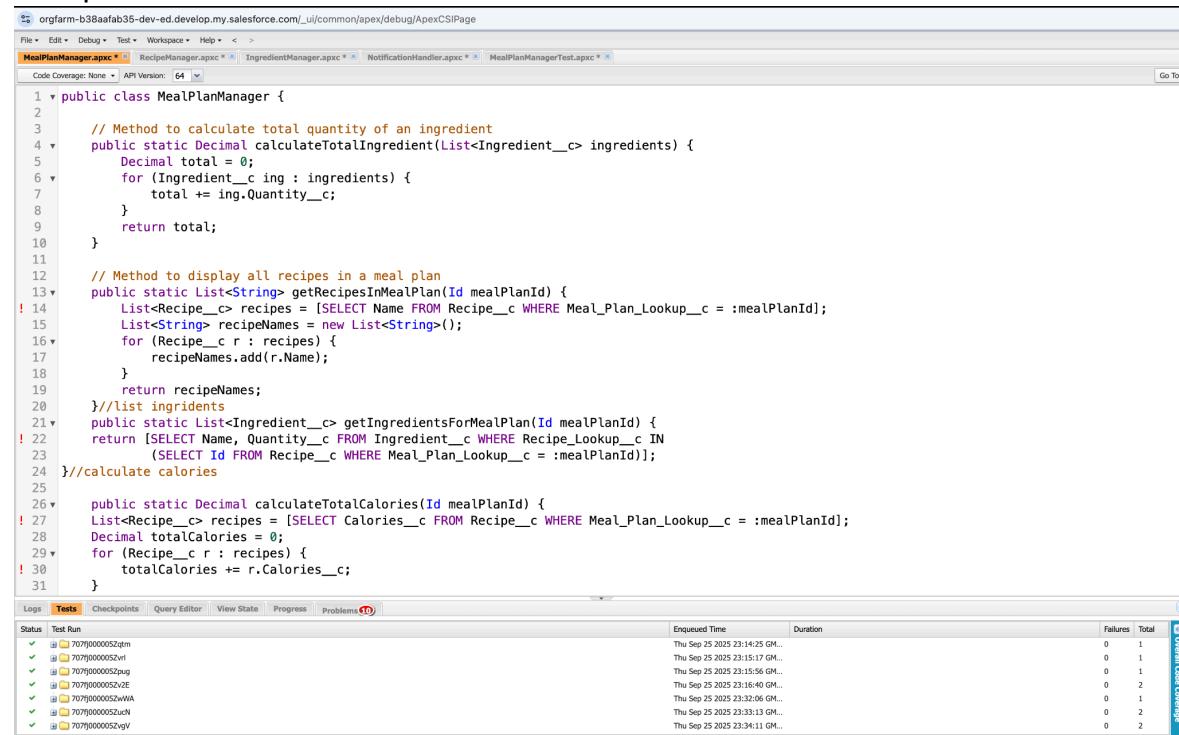
- Meal_Plan__c → Stores details of each meal plan.
- Recipe__c → Stores recipes linked to meal plans.
- Ingredient__c → Stores ingredients with quantity and calories.

Classes:

Classes in Salesforce are written in **Apex** (Salesforce's programming language).

They define **behavior (logic)** that works with objects.

A class contains **methods** (functions) to perform operations like calculations, updates, and queries.



```

1  public class MealPlanManager {
2
3      // Method to calculate total quantity of an ingredient
4      public static Decimal calculateTotalIngredient(List<Ingredient__c> ingredients) {
5          Decimal total = 0;
6          for (Ingredient__c ing : ingredients) {
7              total += ing.Quantity__c;
8          }
9          return total;
10     }
11
12     // Method to display all recipes in a meal plan
13     public static List<String> getRecipesInMealPlan(Id mealPlanId) {
14         List<Recipe__c> recipes = [SELECT Name FROM Recipe__c WHERE Meal_Plan_Lookup__c = :mealPlanId];
15         List<String> recipeNames = new List<String>();
16         for (Recipe__c r : recipes) {
17             recipeNames.add(r.Name);
18         }
19         return recipeNames;
20     }
21     //list ingredients
22     public static List<Ingredient__c> getIngredientsForMealPlan(Id mealPlanId) {
23         return [SELECT Name, Quantity__c FROM Ingredient__c WHERE Recipe_Lookup__c IN
24                 (SELECT Id FROM Recipe__c WHERE Meal_Plan_Lookup__c = :mealPlanId)];
25     }
26     //calculate calories
27     public static Decimal calculateTotalCalories(Id mealPlanId) {
28         List<Recipe__c> recipes = [SELECT Calories__c FROM Recipe__c WHERE Meal_Plan_Lookup__c = :mealPlanId];
29         Decimal totalCalories = 0;
30         for (Recipe__c r : recipes) {
31             totalCalories += r.Calories__c;
32         }
33     }
34 }
```

Status	Test Run	Duration	Failures	Total
✓	70790000052zdm	Thu Sep 25 2025 23:14:25 GM...	0	1
✓	70790000052vrl	Thu Sep 25 2025 23:15:17 GM...	0	1
✓	70790000052zsp	Thu Sep 25 2025 23:15:56 GM...	0	1
✓	70790000052vZE	Thu Sep 25 2025 23:16:40 GM...	0	2
✓	70790000052wWA	Thu Sep 25 2025 23:32:06 GM...	0	1
✓	70790000052xN	Thu Sep 25 2025 23:33:13 GM...	0	2
✓	70790000052vgV	Thu Sep 25 2025 23:34:11 GM...	0	2

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

MealPlanManager.apxc * RecipeManager.apxc * NotificationHandler.apxc * MealPlanManagerTest.apxc *

Code Coverage: None ▾ API Version: 64 ▾ Go To

```

1 * public class RecipeManager {
2 *     public static Decimal calculateTotalCalories(Id recipeId) {
3 *         Decimal totalCalories = 0;
4 *         for (Ingredient__c ing : [SELECT Calories__c FROM Ingredient__c WHERE Recipe_Lookup__c = :recipeId]) {
5 *             totalCalories += (ing.Calories__c == null ? 0 : ing.Calories__c);
6 *         }
7 *         return totalCalories;
8 *     }
9 * }
10

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	70790000052qtm	Thu Sep 25 2025 23:14:25 GM...		0	1
✓	70790000052vrl	Thu Sep 25 2025 23:15:17 GM...		0	1
✓	70790000052puq	Thu Sep 25 2025 23:15:56 GM...		0	1
✓	70790000052wot	Thu Sep 25 2025 23:16:46 GM...		0	1
✓	70790000052wka	Thu Sep 25 2025 23:32:06 GM...		0	2
✓	70790000052ucl	Thu Sep 25 2025 23:33:13 GM...		0	2
✓	70790000052vgv	Thu Sep 25 2025 23:34:11 GM...		0	2

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

MealPlanManager.apxc * RecipeManager.apxc * IngredientManager.apxc * NotificationHandler.apxc * MealPlanManagerTest.apxc *

Code Coverage: None ▾ API Version: 64 ▾ Go To

```

1 * public class IngredientManager {
2 *     public static Decimal calculateTotalQuantity(List<Ingredient__c> ingredients) {
3 *         Decimal total = 0;
4 *         for (Ingredient__c ing : ingredients) {
5 *             total += (ing.Quantity__c == null ? 0 : ing.Quantity__c);
6 *         }
7 *         return total;
8 *     }
9 * }
10

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	70790000052qtm	Thu Sep 25 2025 23:14:25 GM...		0	1
✓	70790000052zg	Thu Sep 25 2025 23:15:17 GM...		0	1
✓	70790000052puq	Thu Sep 25 2025 23:15:56 GM...		0	1
✓	70790000052wot	Thu Sep 25 2025 23:16:46 GM...		0	1
✓	70790000052wka	Thu Sep 25 2025 23:32:06 GM...		0	2
✓	70790000052ucl	Thu Sep 25 2025 23:33:13 GM...		0	2
✓	70790000052vgv	Thu Sep 25 2025 23:34:11 GM...		0	2

orgfarm-b38aab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ >

MealPlanManager.apxc * RecipeManager.apxc * IngredientManager.apxc * NotificationHandler.apxc * MealPlanManagerTest.apxc *

Code Coverage: None ▾ API Version: 64 ▾ Go To

```

1 *public class IngredientManager {
2     public static Decimal calculateTotalQuantity(List<Ingredient__c> ingredients) {
3         Decimal total = 0;
4         for (Ingredient__c ing : ingredients) {
5             total += (ing.Quantity__c == null ? 0 : ing.Quantity__c);
6         }
7         return total;
8     }
9 }
10

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7079000005Zqtm	Thu Sep 25 2025 22:14:25 GM...		0	1	
✓	7079000005Zvrl	Thu Sep 25 2025 22:15:17 GM...		0	1	
✓	7079000005Zpug	Thu Sep 25 2025 22:15:56 GM...		0	1	
✓	7079000005Zv2E	Thu Sep 25 2025 22:16:40 GM...		0	2	
✓	7079000005ZuN	Thu Sep 25 2025 22:32:09 GM...		0	1	
✓	7079000005ZvqV	Thu Sep 25 2025 22:33:13 GM...		0	2	
✓	7079000005ZvqV	Thu Sep 25 2025 22:34:11 GM...		0	2	

orgfarm-b38aab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ >

MealPlanManager.apxc * RecipeManager.apxc * IngredientManager.apxc * NotificationHandler.apxc * MealPlanManagerTest.apxc *

Code Coverage: None ▾ API Version: 64 ▾ Go To

```

1 *public class IngredientManager {
2     public static Decimal calculateTotalQuantity(List<Ingredient__c> ingredients) {
3         Decimal total = 0;
4         for (Ingredient__c ing : ingredients) {
5             total += (ing.Quantity__c == null ? 0 : ing.Quantity__c);
6         }
7         return total;
8     }
9 }
10

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7079000005Zqtm	Thu Sep 25 2025 23:14:25 GM...		0	1	
✓	7079000005Zvrl	Thu Sep 25 2025 23:15:17 GM...		0	1	
✓	7079000005Zpug	Thu Sep 25 2025 23:15:56 GM...		0	1	
✓	7079000005Zv2E	Thu Sep 25 2025 23:16:40 GM...		0	2	
✓	7079000005ZuN	Thu Sep 25 2025 23:32:09 GM...		0	1	
✓	7079000005ZvqV	Thu Sep 25 2025 23:33:13 GM...		0	2	
✓	7079000005ZvqV	Thu Sep 25 2025 23:34:11 GM...		0	2	

orgfarm-b3Baafab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File • Edit • Debug • Test • Workspace • Help • < >

MealPlanManager.apxc * RecipeManager.apxc * IngredientManager.apxc * NotificationHandler.apxc * MealPlanManagerTest.apxc *

Code Coverage: None • API Version: 64

```

1 public class NotificationHandler {
2     public static void sendMealPlanNotification(Set<Id> mealPlanIds) {
3         if (mealPlanIds == null || mealPlanIds.isEmpty()) return;
4         List<Meal_Plan__c> mps = [SELECT Id, Name FROM Meal_Plan__c WHERE Id IN :mealPlanIds];
5         List<Messaging.SingleEmailMessage> mails = new List<Messaging.SingleEmailMessage>();
6         String baseUrl = URL.getSalesforceBaseUrl().toExternalForm();
7         for (Meal_Plan__c mp : mps) {
8             Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
9             // TODO: replace with appropriate recipient logic (owner email, custom setting, etc.)
10            mail.setToAddresses(new String[] { 'vigna.bavares07@gmail.com' });
11            mail.setSubject('New Meal Plan Created: ' + mp.Name);
12            mail.setPlainTextBody('A new Meal Plan "' + mp.Name + '" has been created. Open: ' + baseUrl + '/' + mp.Id);
13            mails.add(mail);
14        }
15        if (!mails.isEmpty()) {
16            Messaging.sendEmail(mails);
17        }
18    }
19 }
20

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7079000005Zqtm	Thu Sep 25 2025 23:14:25 GM...		0	1	
✓	7079000005Zvrl	Thu Sep 25 2025 23:15:17 GM...		0	1	
✓	7079000005Zvg	Thu Sep 25 2025 23:15:56 GM...		0	1	
✓	7079000005ZvZE	Thu Sep 25 2025 23:16:00 GM...		0	2	
✓	7079000005ZwWA	Thu Sep 25 2025 23:22:06 GM...		0	1	
✓	7079000005ZwN	Thu Sep 25 2025 23:32:13 GM...		0	2	
✓	7079000005ZvgV	Thu Sep 25 2025 23:34:11 GM...		0	2	

orgfarm-b3Baafab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File • Edit • Debug • Test • Workspace • Help • < >

MealPlanManager.apxc * RecipeManager.apxc * IngredientManager.apxc * NotificationHandler.apxc * MealPlanManagerTest.apxc *

Code Coverage: None • API Version: 64

```

1 @isTest
2 public class MealPlanManagerTest {
3
4     @isTest
5     static void testCalculateTotalIngredient() {
6         // Create dummy Ingredients
7         List<Ingredient__c> ingredients = new List<Ingredient__c>();
8         ingredients.add(new Ingredient__c(Quantity__c = 2));
9         ingredients.add(new Ingredient__c(Quantity__c = 3));
10
11        Decimal total = MealPlanManager.calculateTotalIngredient(ingredients);
12        System.assertEquals(5, total, 'Total quantity should be 5');
13    }
14
15    @isTest
16    static void testGetRecipesInMealPlan() {
17        // Create a dummy Meal Plan
18        Meal_Plan__c mealPlan = new Meal_Plan__c(Name = 'Test Plan');
19        insert mealPlan;
20
21        // Create Recipes linked to it
22        Recipe__c r1 = new Recipe__c(Name = 'Recipe A', Meal_Plan_Lookup__c = mealPlan.Id);
23        Recipe__c r2 = new Recipe__c(Name = 'Recipe B', Meal_Plan_Lookup__c = mealPlan.Id);
24        insert new List<Recipe__c>{r1, r2};
25
26        List<String> recipes = MealPlanManager.getRecipesInMealPlan(mealPlan.Id);
27        System.assertEquals(2, recipes.size(), 'Meal plan should return 2 recipes');
28    }
29
30

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7079000005Zqtm	Thu Sep 25 2025 23:14:25 GM...		0	1	
✓	7079000005Zvrl	Thu Sep 25 2025 23:15:17 GM...		0	1	
✓	7079000005Zvg	Thu Sep 25 2025 23:15:56 GM...		0	1	
✓	7079000005ZvZE	Thu Sep 25 2025 23:16:40 GM...		0	2	
✓	7079000005ZwWA	Thu Sep 25 2025 23:32:06 GM...		0	1	
✓	7079000005ZwN	Thu Sep 25 2025 23:33:13 GM...		0	2	
✓	7079000005ZvgV	Thu Sep 25 2025 23:34:11 GM...		0	2	

Apex Triggers (before/after insert/update/delete)

What is an Apex Trigger?

- A **trigger** is a piece of code that executes **before or after a record is inserted, updated, deleted, or undeleted.**
- Triggers allow you to automate **business logic** that cannot be done using workflow rules or Process Builder alone.
- **Types:**
 - **Before triggers:** Execute **before** a record is saved to the database. Used for validation or modifying record values.
 - **After triggers:** Execute **after** a record is saved. Used for actions that need record IDs or related objects.

The screenshot shows the Salesforce developer console interface. At the top, the URL is orgfarm-b38aaafab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The tabs at the top include File, Edit, Debug, Test, Workspace, Help, and several open files: MealPlanManager.apxc, RecipeManager.apxc, IngredientManager.apxc, NotificationHandler.apxc, MealPlanManagerTest.apxc, MealPlanTrigger.apxt (highlighted in orange), and MealPlanTriggerTest.apxt. Below the tabs, there's a dropdown for Code Coverage: None and API Version: 64. The main area contains the Apex trigger code:

```
1 v trigger MealPlanTrigger on Meal_Plan__c (before insert, before update) {
2 v   for (Meal_Plan__c mp : Trigger.new) {
3   // Example logic: set default status
! 4 v   if(mp.Status__c == null) {
! 5   mp.Status__c = 'Pending';
! 6 }
! 7 }
! 8 }
! 9 }
! 10
```

At the bottom, there's a navigation bar with Log, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Tests tab is selected. The Test Run table shows the following data:

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7079000005zv2E	Thu Sep 25 2025 23:16:40 GM...		0	2
✓	7079000005zvWA	Thu Sep 25 2025 23:32:06 GM...		0	1
✓	7079000005zvch	Thu Sep 25 2025 23:33:13 GM...		0	2
✓	7079000005zvgV	Thu Sep 25 2025 23:34:11 GM...		0	2
✓	7079000005zj8	Thu Sep 25 2025 23:42:27 GM...		0	1
✓	7079000005zwfs	Thu Sep 25 2025 23:51:38 GM...		0	1

On the far right of the table, there's an "Overall Code Coverage" link.

```

1  @isTest
2  public class MealPlanTriggerTest {
3
4    @isTest static void testBeforeInsert() {
5      // Create test record
6      Meal_Plan__c mp = new Meal_Plan__c(
7        Meal_Plan_Name__c = 'Weekly Healthy Plan',
8        Start_Date__c = Date.today(),
9        End_Date__c = Date.today().addDays(7)
10     );
11
12     // Insert triggers "before insert"
13     insert mp;
14
15     // Verify if trigger logic executed
16     System.assertEquals('Pending', mp.Status__c);
17   }
18
19   @isTest static void testBeforeUpdate() {
20     Meal_Plan__c mp = new Meal_Plan__c(
21       Meal_Plan_Name__c = 'Weekly Healthy Plan',
22       Start_Date__c = Date.today(),
23       End_Date__c = Date.today().addDays(7)
24     );
25     insert mp;
26
27     // Update triggers "before update"
28     mp.End_Date__c = Date.today().addDays(10);
29     update mp;
30
31     // Verify trigger worked
32     System.assert(mp.End_Date__c > mp.Start_Date__c);
}

```

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
✓	7079j0000005zv2E	Thu Sep 25 2025 23:16:40 GM...		0	2	
✓	7079j0000005zwA	Thu Sep 25 2025 23:32:06 GM...		0	1	
✓	7079j0000005zvN	Thu Sep 25 2025 23:33:13 GM...		0	2	
✓	7079j0000005zvqV	Thu Sep 25 2025 23:34:11 GM...		0	2	
✓	7079j0000005zvJ8	Thu Sep 25 2025 23:42:27 GM...		0	1	
✓	7079j0000005zwf5	Thu Sep 25 2025 23:51:38 GM...		0	1	

Trigger Design Pattern

Trigger Design Pattern in Salesforce is used to organize and maintain Apex triggers efficiently. Instead of writing all logic in the trigger itself, a **Handler Class** is created to manage the actual logic for before or after operations. This pattern ensures:

- Separation of concerns (trigger only calls handler methods).
 - Easy maintenance and testing.
 - Scalability and bulk-safe operations.
- Triggers are lightweight and delegate business logic to handler classes, improving code quality and governance.

SOQL & SOSL

- **SOQL (Salesforce Object Query Language):**

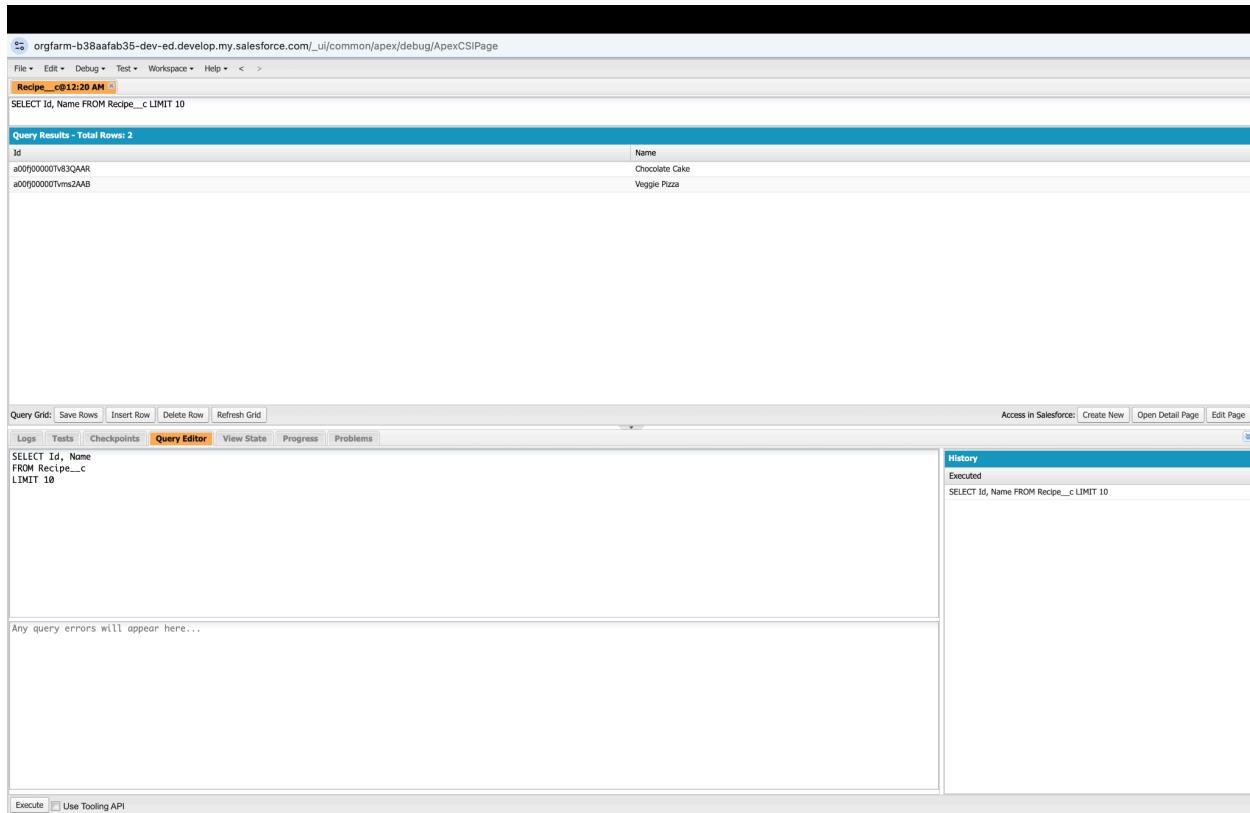
Used to **query records from a single object or related objects.**

Example: Get all Meal Plans with a certain name.

- **SOSL (Salesforce Object Search Language):**

Used to **search across multiple objects and fields.**

Example: Find any record (Meal Plan, Recipe, Ingredient) containing "Salad".



The screenshot shows the Salesforce Debug Console interface. At the top, there's a navigation bar with links like File, Edit, Debug, Test, Workspace, Help, and a timestamp of 12:20 AM. Below the navigation is a query editor window with the following content:

```
SELECT Id, Name FROM Recipe__c LIMIT 10
```

Underneath the query editor is a table titled "Query Results - Total Rows: 2". The table has two columns: "Id" and "Name". The data rows are:

Id	Name
a00fj00000Tv83QAR	Chocolate Cake
a00fj00000Tvms2AAB	Veggie Pizza

At the bottom of the query editor, there are buttons for "Logs", "Tests", "Checkpoints", "Query Editor" (which is highlighted in orange), "View State", "Progress", and "Problems". To the right of the query editor is a "History" panel which also displays the executed query:

```
SELECT Id, Name FROM Recipe__c LIMIT 10
```

For SOQL

The screenshot shows the Salesforce Debug Console interface. At the top, there's a navigation bar with links like File, Edit, Debug, Test, Workspace, Help, and a status message "Log executeAnonymous @26/09/2025, 00:26:43". Below the navigation is a "Logs" section titled "Execution Log" with columns for Timestamp, Event, and Details. A log entry from "00:26:43:222" shows a USER_DEBUG log: "[6]DEBUG(((),))". The main area contains a "Query Editor" tab where the query "SELECT Id, Name FROM Recipe__c LIMIT 10" is typed. To the right of the editor is a "History" panel showing the same query. At the bottom, there are buttons for Execute and Use Tooling API.

For SOSL

Collections: List, Set, Map

What Are Collections?

Collections in Apex are **data structures** that allow you to store multiple values in a single variable. There are **3 main types**:

1. **List** – An ordered collection of elements. Duplicates allowed.
2. **Set** – An unordered collection of **unique** elements. No duplicates.
3. **Map** – A collection of **key-value pairs**. Each key is unique.

```

1 v public class RecipeCollections {
2
3     // [List Example: Get all recipes and display names
4     public static void listAllRecipes() {
5         List<Recipe__c> recipes = [SELECT Id, Name FROM Recipe__c];
6         System.debug('--- List of Recipes ---');
7         for(Recipe__c r : recipes) {
8             System.debug('Recipe ID: ' + r.Id + ', Name: ' + r.Name);
9         }
10    }
11
12    // [Set Example: Get unique recipe names
13    public static void setUniqueRecipeNames() {
14        Set<String> recipeNames = new Set<String>();
15        List<Recipe__c> recipes = [SELECT Name FROM Recipe__c];
16        for(Recipe__c r : recipes) {
17            recipeNames.add(r.Name);
18        }
19        System.debug('--- Unique Recipe Names ---');
20        for(String nameVal : recipeNames) {
21            System.debug(nameVal);
22        }
23    }
24
25    // [Map Example: Map recipe ID to recipe name
26    public static void mapRecipeIdToName() {
27        Map<Id, String> recipeMap = new Map<Id, String>();
28        List<Recipe__c> recipes = [SELECT Id, Name FROM Recipe__c];
29        for(Recipe__c r : recipes) {
30            recipeMap.put(r.Id, r.Name);
31        }
32    }
}

```

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

```

SELECT Id, Name
FROM Recipe__c
LIMIT 10

```

Any query errors will appear here...

History
Executed
SELECT Id, Name FROM Recipe__c LIMIT 10

Execute Use Tooling API

Control Statements

Control statements allow you to control the **flow of execution** in Apex code. These include:

1. **Decision-making statements:** if, else if, else, switch.
2. **Loops:** for, while, do-while.
3. **Jump statements:** break, continue, return.

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

Recipe__c@12:20 AM Log executeAnonymous @26/09/2025, 00:26:43 RecipeCollections.apxc Log executeAnonymous @26/09/2025, 00:36:03 ControlStatementsDemo.apxc Log executeAnonymous @26/09/2025, 00:48:42

Code Coverage: None API Version: 64 Go To

```

1 v public class ControlStatementsDemo {
2
3 v     public static void checkRecipeNameLength() {
4         List<Recipe__c> recipes = [SELECT Id, Name FROM Recipe__c LIMIT 5];
5         for(Recipe__c r : recipes) {
6             if(r.Name.length() > 10) {
7                 System.debug('Long recipe name: ' + r.Name);
8             } else if(r.Name.length() > 5) {
9                 System.debug('Medium recipe name: ' + r.Name);
10            } else {
11                System.debug('Short recipe name: ' + r.Name);
12            }
13        }
14    }
15
16    public static void loopRecipes() {
17        List<Recipe__c> recipes = [SELECT Id, Name FROM Recipe__c LIMIT 10];
18        System.debug('--- For Loop ---');
19        for(Recipe__c r : recipes) {
20            if(r.Name.startsWith('A')) continue;
21            System.debug('Recipe Name: ' + r.Name);
22            if(r.Name.endsWith('e')) break;
23        }
24
25        System.debug('--- While Loop ---');
26        Integer i = 0;
27        while(i < recipes.size()) {
28            System.debug('Recipe Id: ' + recipes[i].Id);
29        }
30    }
}

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name	Line	Problem
MealPlanCollections	5	SELECT Id, Meal_Plan_Name__c FROM Meal_Plan__c ^ ERROR at Row:1:Column:12 No such column 'Meal_Plan_Name__c' on entity 'Meal_Plan__c'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference your Apex code for examples.
MealPlanCollections	9	Variable does not exist: Meal_Plan_Name__c

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

Recipe__c@12:20 AM Log executeAnonymous @26/09/2025, 00:26:43 RecipeCollections.apxc Log executeAnonymous @26/09/2025, 00:36:03 ControlStatementsDemo.apxc Log executeAnonymous @26/09/2025, 00:48:42

Execution Log

Timestamp	Event	Details
00:48:42:001	USER_INFO	[EXTERNAL]@005fj00000sP0d vigna.bavana07955@agentforce.com (GMT-07:00) Pacific Daylight Time (America/Los_Angeles) GMT-07:00
00:48:42:001	EXECUTION_STARTED	
00:48:42:001	CODE_UNIT_STARTED [EXTERNAL]execute_anonymous_apex	
00:48:42:001	HEAP_ALLOCATE	[95]Bytes:3
00:48:42:001	HEAP_ALLOCATE	[100]Bytes:152
00:48:42:001	HEAP_ALLOCATE	[417]Bytes:408
00:48:42:001	HEAP_ALLOCATE	[430]Bytes:408
00:48:42:001	HEAP_ALLOCATE	[317]Bytes:6
00:48:42:001	HEAP_ALLOCATE	[EXTERNAL]Bytes:1
00:48:42:001	STATEMENT_EXECUTE	[1]
00:48:42:001	STATEMENT_EXECUTE	[1]
00:48:42:001	HEAP_ALLOCATE	[68]Bytes:5
00:48:42:001	HEAP_ALLOCATE	[74]Bytes:5
00:48:42:001	HEAP_ALLOCATE	[82]Bytes:7
00:48:42:001	SYSTEM_MODE	false
00:48:42:001	HEAP_ALLOCATE	[1]Bytes:5
00:48:42:013	HEAP_ALLOCATE	[1]Bytes:7
00:48:42:013	HEAP_ALLOCATE	[1]Bytes:33
00:48:42:013	METHOD_ENTRY	[1]@01pfj000003pN0r ControlStatementsDemo.ControlStatementsDemo()
00:48:42:013	STATEMENT_EXECUTE	[1]
00:48:42:013	STATEMENT_EXECUTE	[1]
00:48:42:013	METHOD_EXIT	[1]@01pfj000003pN0r ControlStatementsDemo
00:48:42:013	METHOD_ENTRY	[1]@01pfj000003pN0r ControlStatementsDemo.CheckRecipeNameLength()
00:48:42:013	STATEMENT_EXECUTE	[3]
00:48:42:013	STATEMENT_EXECUTE	[4]
00:48:42:013	HEAP_ALLOCATE	[4]Bytes:38
00:48:42:013	HEAP_ALLOCATE	[4]Bytes:4
00:48:42:014	SOQL_EXECUTE	[4]Aggregations:0 SELECT Id, Name FROM Recipe__c LIMIT 5
00:48:42:023	SOQL_EXECUTE	[4]Rows:2
00:48:42:023	HEAP_ALLOCATE	[4]Bytes:12
00:48:42:023	HEAP_ALLOCATE	[4]Bytes:12

This Frame Executable Debug Only Filter Click here to filter the log

Logs Tests Checkpoints Query Editor View State Progress Problems

Name	Line	Problem
MealPlanCollections	5	SELECT Id, Meal_Plan_Name__c FROM Meal_Plan__c ^ ERROR at Row:1:Column:12 No such column 'Meal_Plan_Name__c' on entity 'Meal_Plan__c'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference your Apex code for examples.
MealPlanCollections	9	Variable does not exist: Meal_Plan_Name__c

Batch Apex

Batch Apex is used to process **large amounts of records** asynchronously (in the background).

Each batch executes a subset of records (called **batch size**), preventing governor limit issues.

It implements the Database.Batchable<SObject> interface.

```
1 public class RecipeBatchProcessor implements Database.Batchable<SObject> {
2
3     // Start method: Query the records to process
4     public Database.QueryLocator start(Database.BatchableContext BC) {
5         // Get all recipes
6         return Database.getQueryLocator('SELECT Id, Name, Meal_Plan_Lookup__c FROM Recipe__c');
7     }
8
9     // Execute method: Process each batch
10    public void execute(Database.BatchableContext BC, List<Recipe__c> scope) {
11        for(Recipe__c r : scope) {
12            System.debug('Processing Recipe: ' + r.Name);
13            // Optional: Perform some updates
14            // r.Some_Field__c = 'Updated';
15        }
16        // Optional: Update records if you changed them
17        // update scope;
18    }
19
20    // Finish method: Actions after all batches processed
21    public void finish(Database.BatchableContext BC) {
22        System.debug('Batch Apex finished processing recipes.');
23    }
24}
25
```

Enter Apex Code

```
1 // Create an instance of the batch class
2 RecipeBatchProcessor batchInstance = new RecipeBatchProcessor();
3 // Execute the batch with a batch size of 200
4 Database.executeBatch(batchInstance, 200);
```

Open Log Execute Execute Highlighted

Name	Line	Problem
MealPlanCollections	5	SELECT Id, Meal_Plan_Name__c FROM Meal_Plan__c ^ ERROR at Row:1:Column:12 No such column 'Meal_Plan_Name__c' on entity 'Meal_Plan__C'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference y...
MealPlanCollections	9	Variable does not exist: Meal_Plan_Name__c

Queueable Apex:

Queueable Apex is a Salesforce feature that allows you to run **asynchronous operations** (background jobs) in a **flexible and scalable** way. It is similar to Batch Apex but simpler for smaller tasks and allows **chaining jobs**.

orgfarm-b38aafab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

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

RecipeCollections.apxc Log executeAnonymous @26/09/2025, 00:36:03 ControlStatementsDemo.apxc Log executeAnonymous @26/09/2025, 00:48:42 RecipeBatchProcessor.apxc RecipeQueueProcessor.apxc Log executeAnonymous @26/09/2025, 01:02:34 Go To

Code Coverage: None API Version: 64 Go To

```

1 v public class RecipeQueueProcessor implements Queueable {
2
3 v     public void execute(ExecutionContext context) {
4         // Query all recipes for Id and Name only
5         List<Recipe__c> recipes = [SELECT Id, Name FROM Recipe__c];
6
7         // Process each recipe
8         for (Recipe__c r : recipes) {
9             System.debug('Processing Recipe: Id = ' + r.Id + ', Name = ' + r.Name);
10        }
11    }
12 }
13

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name	Line	Problem
MealPlanCollections	5	SELECT Id, Meal_Plan_Name__c FROM Meal_Plan__c ^ ERROR at Row:1:Column:12 No such column 'Meal_Plan_Name__c' on entity 'Meal_Plan__c'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference y...
MealPlanCollections	9	Variable does not exist: Meal_Plan_Name__c

orgfarm-b38aafab35-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

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

RecipeCollections.apxc Log executeAnonymous @26/09/2025, 00:36:03 ControlStatementsDemo.apxc Log executeAnonymous @26/09/2025, 00:48:42 RecipeBatchProcessor.apxc RecipeQueueProcessor.apxc Log executeAnonymous @26/09/2025, 01:02:34

Execution Log

Timestamp	Event	Details
01:02:34:001	CODE_UNIT_STARTED [EXTERNAL]::execute_anonymous_apex	
01:02:34:001	STATEMENT_EXECUTE [1]	
01:02:34:001	STATEMENT_EXECUTE [1]	
01:02:34:014	METHOD_ENTRY [1][0]@fj0000003p13[RecipeQueueProcessor:RecipeQueueProcessor]	
01:02:34:014	STATEMENT_EXECUTE [1]	
01:02:34:014	STATEMENT_EXECUTE [1]	
01:02:34:014	METHOD_EXIT [1][0]@fj0000003p13[RecipeQueueProcessor]	
01:02:34:014	CONSTRUCTOR [1][0]@fj0000003p13<init>()	RecipeQueueProcessor
01:02:34:014	VARIABLE_ASSIGN [1][0]@fj0000003p13[0x794d992]	
01:02:34:016	STATEMENT_EXECUTE [1]	
01:02:34:016	CONSTRUCTOR [1][0]@fj0000003p13<init>()	RecipeQueueProcessor
01:02:34:042	CODE_UNIT_FINISHED execute_anonymous_apex	

This Frame Executable Debug Only Filter Click here to filter the log

Logs Tests Checkpoints Query Editor View State Progress Problems

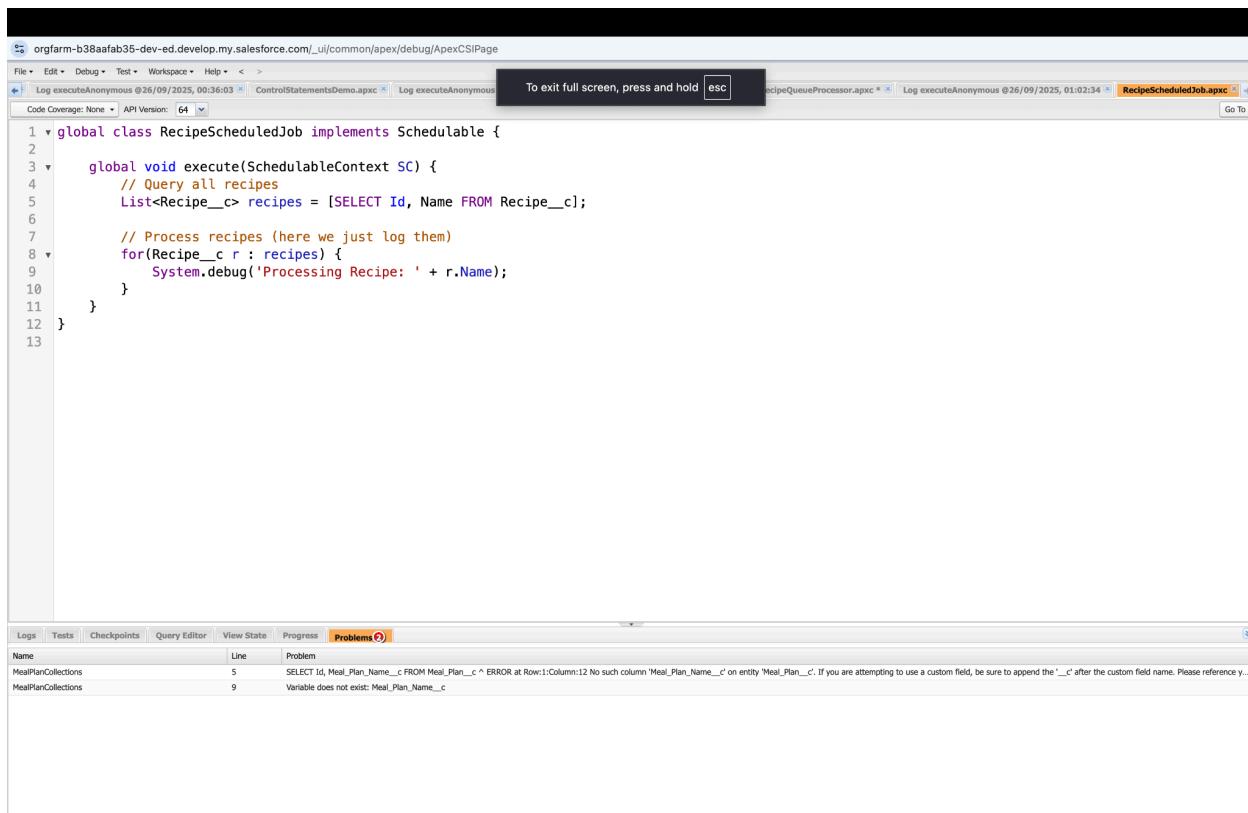
Name	Line	Problem
MealPlanCollections	5	SELECT Id, Meal_Plan_Name__c FROM Meal_Plan__c ^ ERROR at Row:1:Column:12 No such column 'Meal_Plan_Name__c' on entity 'Meal_Plan__c'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference y...
MealPlanCollections	9	Variable does not exist: Meal_Plan_Name__c

Scheduled Apex:

Scheduled Apex allows you to run **Apex classes at specified times** or on a recurring schedule. It is ideal for **automated, time-based operations** in Salesforce.

Use Case:

- Automatically update recipe records daily.
- Send email notifications at a scheduled time.
- Perform cleanup tasks or batch processing without manual intervention.



The screenshot shows the Salesforce Apex code editor interface. The main pane displays the code for a global class named `RecipeScheduledJob` that implements the `Schedulable` interface. The code queries all recipes and processes them by logging their names. A tooltip indicates that pressing `esc` will exit full screen mode. The bottom pane shows the `Problems` tab with one error: a column named `Meal_Plan__c` does not exist in the `Meal_Plan__c` entity, suggesting a misspelling of the custom field name.

```
1 v global class RecipeScheduledJob implements Schedulable {  
2  
3 v     global void execute(SchedulableContext SC) {  
4         // Query all recipes  
5         List<Recipe__c> recipes = [SELECT Id, Name FROM Recipe__c];  
6  
7         // Process recipes (here we just log them)  
8         for(Recipe__c r : recipes) {  
9             System.debug('Processing Recipe: ' + r.Name);  
10        }  
11    }  
12 }
```

Name	Line	Problem
MealPlanCollections	5	SELECT Id, Meal_Plan__c FROM Meal_Plan__c ^ ERROR at Row:1:Column:12 No such column 'Meal_Plan__c' on entity 'Meal_Plan__c'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference yo...
MealPlanCollections	9	Variable does not exist: Meal_Plan__c

The screenshot shows the Salesforce Apex Classes page. At the top, there's a message about Apex usage: "Percent of Apex Used: 0.0% You are currently using 3,926 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." Below this, there's a "Compile all classes" button and a "View" dropdown.

The main area displays a table of Apex classes:

Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit Del Security	ControlStatementsDemo		64.0	Active	1,700	vigna_banana	9/25/2025, 12:18 PM
Edit Del Security	DeveloperEditionUtils	devedapp	64.0	Active	164	OrgFarm EPIC	9/19/2025, 4:35 AM
Edit	DeveloperEditionUtilsTest	devedapp	64.0	Active	261	OrgFarm EPIC	9/19/2025, 4:35 AM
Edit Del Security	IngredientManager		64.0	Active	35	vigna_banana	9/25/2025, 10:45 AM
Edit Del Security	MealPlanCollections		64.0	Active	37	vigna_banana	9/25/2025, 12:00 PM
Edit Del Security	MealPlanManager		64.0	Active	33	vigna_banana	9/25/2025, 10:38 AM
Edit Del Security	MealPlanManagerTest		64.0	Active	37	vigna_banana	9/25/2025, 11:01 AM
Edit Del Security	NotificationHandler		64.0	Active	37	vigna_banana	9/25/2025, 10:46 AM
Edit Del Security	PostInstallScript	devedapp	64.0	Active	2,175	OrgFarm EPIC	9/19/2025, 4:35 AM
Edit	PostInstallScriptTest	devedapp	64.0	Active	781	OrgFarm EPIC	9/19/2025, 4:35 AM
Edit Del Security	RecipeBatchProcessor		64.0	Active	38	vigna_banana	9/25/2025, 12:22 PM
Edit Del Security	RecipeCollections		64.0	Active	1,186	vigna_banana	9/25/2025, 12:08 PM
Edit Del Security	RecipeManager		64.0	Active	31	vigna_banana	9/25/2025, 10:45 AM
Edit Del Security	RecipeQueueProcessor		64.0	Active	201	vigna_banana	9/25/2025, 12:31 PM
Edit Del Security	RecipeScheduledJob		64.0	Active	287	vigna_banana	9/25/2025, 12:43 PM

Below this, there's a section titled "Dynamic Apex Classes" with the subtext "Dynamic Apex extends your programming reach by interacting with Lightning Platform components." It shows a table:

Class Name	Namespace Prefix	Api Version	Created By	Last Modified By
No records to display.				

Future Methods

- **Purpose:** Run long-running operations asynchronously, without blocking the user interface.
- **Key Points:**
 - Annotated with `@future`.
 - Can only accept **primitive data types or collections of primitives** as parameters.
 - Cannot return a value.
- **Example:**

```
public class RecipeAsync {
    @future
    public static void sendRecipeNotification(Id recipeId) {
```

```

        Recipe__c r = [SELECT Name FROM Recipe__c WHERE Id =
:recipeId];
        System.debug('Sending notification for recipe: ' + r.Name);
    }
}

```

Exception Handling

- **Purpose:** Gracefully handle runtime errors in Apex.
- **Key Points:**
 - Use try, catch, and finally blocks.
 - Can throw custom exceptions using throw new CustomException().
- **Example:**

```

public class RecipeHandler {
    public static void updateRecipeCalories(Id recipeId, Decimal
calories) {
        try {
            Recipe__c r = [SELECT Name FROM Recipe__c WHERE Id =
:recipeId];
            r.Calories__c = calories;
            update r;
        } catch (Exception e) {
            System.debug('Error updating recipe: ' + e.getMessage());
        } finally {
            System.debug('Update attempt finished.');
        }
    }
}

```

Test Classes

- **Purpose:** Validate that Apex code works correctly and to meet Salesforce deployment requirements.
- **Key Points:**
 - Annotated with `@isTest`.
 - Create test records, execute logic, and assert outcomes.
 - Must cover at least **75% of code** to deploy to production.
- **Example:**

```
@isTest
public class RecipeTest {
    static testMethod void testRecipeNotification() {
        Recipe__c r = new Recipe__c(Name = 'Test Salad');
        insert r;

        Test.startTest();
        RecipeAsync.sendRecipeNotification(r.Id);
        Test.stopTest();

        System.assert(r.Name == 'Test Salad');
    }
}
```

Asynchronous Processing

- **Purpose:** Handle large operations without timing out.
- **Types in Salesforce:**
 - **Future Methods** → for simple async tasks.
 - **Queueable Apex** → chainable and flexible.

- **Batch Apex** → process large datasets in chunks.
- **Scheduled Apex** → run tasks at specific times.

Key Advantage: Users don't have to wait for the process to finish; Salesforce executes it in the background.