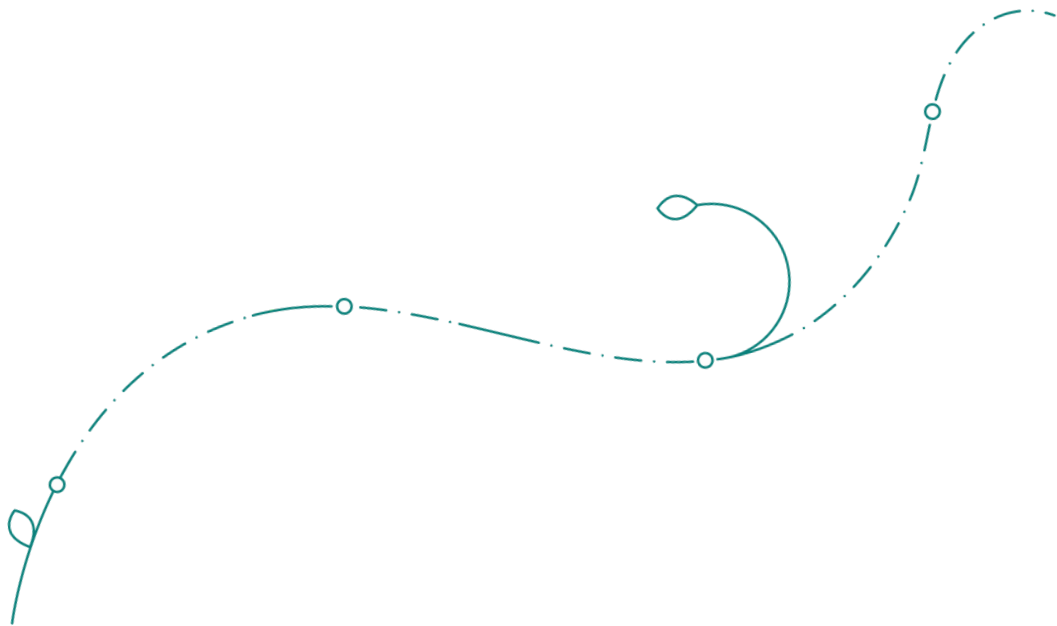


# Flow + Apex + LWC

A Perfect Combination!

EXERCISE WORKBOOK



## Table of Contents

Exercise 1: Populate family lookup field	2
Exercise 2: Populate family lookup field (Fixed)	5
Exercise 3: @invocableMethods with Complex data structures	8
Exercise 4: Display contacts grouped by families	13

### Exercise 1: Populate family lookup field

**Goal:** Attempt to populate the Family\_\_c lookup up field on the Contact record based on the last name

#### What does the flow do?

1. Open the **EX01 - Link family** flow
  - a. What type of flow is this?
    - ☐ French fries flow
    - ☐ Pirate flow
    - ☐ Screen flow
  - Click **Debug**
  - Select **Skip start condition requirements**
  - Select **Updated**
  - Type: **Perez**
  - Select any contact
  - Change Last Name to **Smith**
  - Click **Run**
  - b. Which path did the flow go through?
    - ☐ Family Found
    - ☐ Family must be created
  - Click **Edit Flow**
  - **Activate the flow**

#### How was the solution built?

2. Go back to the **EX01 - Link family** flow
  - Review the **Start** element
  - a. What object is this record-trigger flow firing on?
    - ☐ Account
    - ☐ Contact
    - ☐ Family\_\_c

3. Review the **Get010** element

- a. What is the equivalent SOQL query?

SELECT \_\_\_\_\_  
FROM \_\_\_\_\_  
WHERE \_\_\_\_\_  
ORDER BY \_\_\_\_\_  
LIMIT \_\_\_\_\_

- b. What is the API name of the variable that holds the results? (Hint: Is the same name as the flow element)

4. Review the **IF020** element

- a. What is this condition validating?

- ☐ If the contact has a last name  
☐ If there is a family with this last name

5. Review the **Insert030** element

- a. Which record does it create?

- ☐ Account  
☐ Contact  
☐ Family\_\_c

6. Review the **Update40** element

- a. Which record does it update?

- ☐ Account  
☐ Contact  
☐ Family\_\_c

7. Does it **really** work?

- Open **HOW/main/default/classes/EX01\_LinkFamily\_Test.cls**
- Test methods create contacts with different last name scenarios
- In line #1.5 click **Run All Tests**
- a. Did all the tests succeed?
  - ☐ Yes
  - ☐ No

- In line #25, remove the **comment**
- **Save and deploy**
- In line #23.5 click **Run Test**
- b. Did the tests succeed?
  - ☐ Yes
  - ☐ No

- c. What error did we get?
  - ☐ Governor limit exception
  - ☐ Required field missing
  - ☐ Duplicate values
  - ☐ Other

8. Deactivate flow

- Go back to the flow
- **Deactivate the flow**

### Exercise 2: Populate family lookup field (Fixed)

**Goal:** Use an @invocable Apex method to fix the issue

#### What does the flow do?

1. Open the **EX02 - Link family (Apex)** flow
  - a. What type of flow is this?
    - ☐ French fries flow
    - ☐ Pirate flow
    - ☐ Screen flow
  - Activate this flow
  - Open **HOW/main/default/classes/EX01\_LinkFamily\_Test.cls**
  - In line #23.5 click **Run Test** (*newContacts\_WithDuplicate\_NewLastNames*)
  - b. Did the tests succeed?
    - ☐ Yes
    - ☐ No

#### How was the solution built?

2. Go back to the **EX02 - Link family (Apex)** flow
  - Review the **Apex010** element
  - a. What is the name of the global variable we are sending to Apex?
    - ☐ \$Record
    - ☐ \$Record\_\_prior
    - ☐ Family\_\_c
  - b. How many Contact records does **\$Record** hold?
    - ☐ One
    - ☐ Many

- Expand the **Advanced** section
  - c. What is the name of the variable receiving the results from the Apex call?
    - ☐ contactWithFamily
    - ☐ \$Record\_\_prior
    - ☐ Family\_\_c
  - d. What's the **contactWithFamily** data type?
    - ☐ Contact
    - ☐ List<Contact>
3. Review the **Update020** element
- a. What is the variable name that holds the information we are updating?
    - ☐ contactWithFamily
    - ☐ \$Record\_\_prior
    - ☐ \$Record
4. Review the **EX02\_LinkFamily.cls** Apex class
- Open **HOW/main/default/classes/EX02\_LinkFamily.cls**
  - a. In line #5, which annotation are we using?
    - ☐ @IsTest
    - ☐ @Future
    - ☐ @InvocableMethod
    - ☐ @InvocableVariable
  - b. How many methods annotated with **@InvocableMethod** are in this class?
    - ☐ 1
    - ☐ 2
  - c. In line #6, how many parameters take the method?
    - ☐ 1
    - ☐ 2
  - d. What's the data type of the parameter **contacts**?
    - ☐ Contact
    - ☐ List<Contact>

e. How many Contact records can this variable hold?

- ☐ One
- ☐ Many

f. What's the return type?

- ☐ Contact
- ☐ List<Contact>

g. How many Contact records could be returned?

- ☐ One
- ☐ Many

h. From the flow, we are passing and receiving only one record, but here we are receiving and returning many! **Why?**

i. In line #44, which DML operation are we performing?

- ☐ Insert missing families
- ☐ Update all families
- ☐ Delete old contacts

j. Which line is doing the DML of the contacts? (*tricky*)

- ☐ 17
- ☐ 44
- ☐ None, the DML is performed by the flow

5. Review the permission set

- Open **HOW/main/default/permissionsets/HOW.permissionset-meta.xml**

a. In line #8, which class are we enabling?

- ☐ EX01\_LinkFamily\_Test
- ☐ EX02\_LinkFamily
- ☐ EX03\_QueryFamilies



### Exercise 3: @invocableMethods with Complex data structures

**Goal:** Review how complex data structures work with @invocableMethods

#### What does the flow do?

1. Open the **EX03 - DataStructures** flow
  - a. What type of flow is this?
    - ☐ French fries flow
    - ☐ Pirate flow
    - ☐ Screen flow
  - Open the **Screen030** element
  - b. What is the expression displayed in the display text element?
    - ☐ {!response.theQuestion}
    - ☐ {!response.theAnswer}
  - Click **Cancel**
  - Click **Run** to test this screen flow
  - c. What value is being displayed for **The answer**?
    - ☐ 42
  - d. Is that the right answer? (Geeky joke)
    - ☐ Yes

#### Note:

If you did not understand question 1d, **after class** you must check this [Wikipedia article](#) and this [YouTube video](#)

## How was the solution built?

2. Go back to the **EX03\_DataStructures** flow

- Go back to the **EX03\_DataStructures** flow
- Review the **Get010** element
- a. What is the equivalent SOQL query?

SELECT \_\_\_\_\_  
FROM \_\_\_\_\_  
WHERE \_\_\_\_\_  
ORDER BY \_\_\_\_\_  
LIMIT \_\_\_\_\_

b. What is the API name of the variable that holds the results?

- ☐ Get010
- ☐ Apex020
- ☐ Screen030

3. Review the **Apex020** element

a. How many input values are we sending?

- ☐ 0
- ☐ 2
- ☐ 4

b. What value are we sending for **Family Inputs**?

- ☐ Only one family
- ☐ All the families retrieved by {!Get010}

c. Since **Max Families** value is not required, how can we indicate we are not passing a value?

- ☐ Value is required
- ☐ Leave the textbox empty
- ☐ Just toggle the switch

- Expand the **Advanced** section
  - d. What is the variable name receiving the result from the Apex call?
    - ☐ response
    - ☐ Get010
    - ☐ Apex020
  - e. What is the data type of **response**?
    - ☐ List<EX03AE\_Response>
    - ☐ EX03AE\_Response
    - ☐ List<Family\_\_c>
4. Review the **EX03\_QueryFamilies** Apex class
- Open **HOW/main/default/classes/EX03\_QueryFamilies.cls**
  - a. In line #2, which annotation are we using?
    - ☐ @IsTest
    - ☐ @Future
    - ☐ @InvocableMethod
    - ☐ @InvocableVariable
  - b. What's the data type of the parameter **familyRequests**?
    - ☐ List<FamilyRequest>
    - ☐ FamilyRequest
  - c. In line #12, what does the **familyRequests[0]** mean?
    - ☐ The first Family\_\_c record
    - ☐ The first Flow Interview
    - ☐ The first Flow Transaction
  - d. In line #11, we have an assertion. Will this ever fail? (*tricky*)
    - ☐ Yes, if we have zero Flow Interviews
    - ☐ Yes, pirate flows create Flow Interviews for each record DML
    - ☐ No, this is a screen flow (one and only one Flow Interview)

- e. Where (which line number) are we defining the **FamilyRequest** class?
- ☐ 3
  - ☐ 26
  - ☐ 33
- f. How many variables are defined in the **FamilyRequest** class?
- ☐ 1
  - ☐ 2
  - ☐ 3
- g. Which annotation are we using for them?
- ☐ @IsTest
  - ☐ @Future
  - ☐ @InvocableMethod
  - ☐ @InvocableVariable
- h. What's the **label** for the first variable?
- ☐ Families input
  - ☐ Families output
  - ☐ maxFamilies
- i. What's the **families** data type?
- ☐ Family\_\_c
  - ☐ List<Family\_\_c>
  - ☐ Integer
- j. In line #3, what's the return type for the @invocableMethod?
- ☐ Family\_\_c
  - ☐ List<Family\_\_c>
  - ☐ List<FamilyResponse>
- k. How many elements, and in which order, should that list contain?
- ☐ Must correspond (number, order) to the elements in the parameter
  - ☐ The number and order are not relevant
  - ☐ Any number and order is fine

- l. Which line number is the **FamilyResponse** defined?
  - ☐ 3
  - ☐ 26
  - ☐ 33
  
- m. Which annotation are we using for **response**?
  - ☐ @IsTest
  - ☐ @Future
  - ☐ @InvocableMethod
  - ☐ @InvocableVariable
  
- n. What's the data type of **response**?
  - ☐ FamilyResponse
  - ☐ EX03AE\_Response
  - ☐ EX03\_QueryFamilies
  
- 5. Review the **EX03AE\_Response.cls** Apex class
  - Open **HOW/main/default/classes/EX03AE\_Response.cls**
  - a. How many properties are in the **EX03AE\_Response** class?
    - ☐ 1
    - ☐ 2
    - ☐ 3
  
  - b. Which annotation are we using for them?
    - ☐ @IsTest
    - ☐ @AuraEnabled
    - ☐ @InvocableMethod
    - ☐ @InvocableVariable
  
  - In question 1a, We saw the flow used **{!response.theAnswer}**
    - Thanks, @AuraEnabled!

## Exercise 4: Display contacts grouped by families

**Goal:** Using a LWC display the Family\_\_c records grouped by last name and show their related Contact records

### What does the flow do?

1. Navigate the **Families** application
  - Open the **home** tab
  - Click on the **Finish** button
  - a. What happened?
    - ☐ Error: "You can't go past this screen!"
    - ☐ Navigated to the next screen
    - ☐ Flow completed successfully
  - Find the **Pérez** family
  - Expand the **Pérez** family
  - Click on **Andrés**
  - b. What happened?
    - ☐ Error: "You can't navigate!"
    - ☐ Navigated to the Andrés Pérez contact record
    - ☐ Nothing!

### How was the solution built?

2. Open the **EX04 - TreeGrid** flow
  - a. What type of flow is this?
    - ☐ French fries flow
    - ☐ Pirate flow
    - ☐ Screen flow
  - **NOTE:** This flow is very similar to the previous one, except for the **Screen030** element. So let's review just that.

- Review the **Screen030** element
  - b. What is the name of the custom component?
    - ☐ Display Text
    - ☐ Family Tree Grid
    - ☐ There is none!
  - Select the **Family Tree Grid** component
  - c. What is the label for the textbox containing the expression **{!response}**?
    - ☐ Family Data
    - ☐ API Name
3. Review the **ex04TreeGrid** LWC component
- Find the **HOW/main/default/lwc/ex04TreeGrid** folder
  - Open the **ex04TreeGrid.html** file
  - a. Which standard component are we using?
    - ☐ <lightning-tree-grid />
    - ☐ <lightning-button />
    - ☐ <lightning-card />
  - Open the **ex04TreeGrid.js-meta.xml** file
  - b. In line #5, what is the **masterLabel**?
    - ☐ ex04TreeGrid.js
    - ☐ Undefined
    - ☐ Family Tree Grid
  - c. In line #7, what is the **target**?
    - ☐ lightning\_\_FlowScreen
    - ☐ lightning\_\_HomePage
    - ☐ lightning\_\_RecordPage
  - d. In line #13, what does **name="familyData"** represent?
    - ☐ The JavaScript variable
    - ☐ The label for the property in the screen component
    - ☐ The expected data type that flow should pass

- e. In line #13, what does **label="Family Data"** represent?
  - ☐ The JavaScript variable
  - ☐ The label for the property in the screen component
  - ☐ The expected data type that flow should pass
  
- f. In line #13, what does **type="apex://EX03AE\_Response"** represent?
  - ☐ The JavaScript variable
  - ☐ The label for the property in the screen component
  - ☐ The expected data type that flow should pass
  
- Open the **ex04TreeGrid.js** file
- g. How do we get the families' data?
  - ☐ @wire (data from Apex)
  - ☐ @api familyData (data from the flow)
  - ☐ Imperative Apex (data from Apex)
  
- h. In line #59, what's the error message?
  - ☐ You can't go past this screen!
  - ☐ You can't view contacts
  
- i. In line #77, what are we doing?
  - ☐ Navigating to the Contact record
  - ☐ Displaying an error message