TrailblazerDX Hands-on Workshop

Flow + Apex + LWC

A Perfect Combination!

EXERCISE WORKBOOK

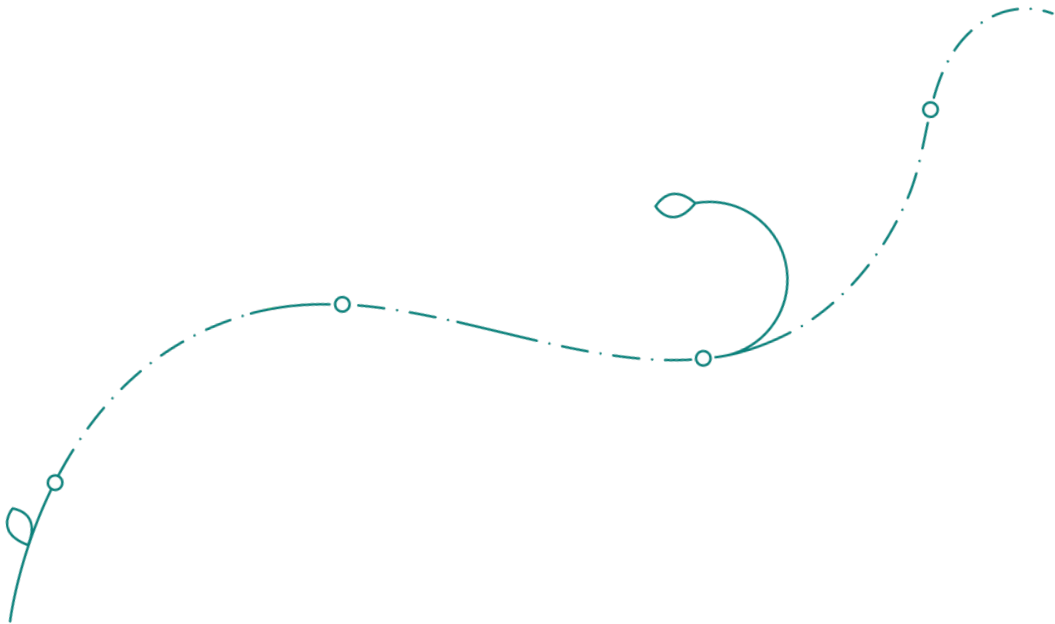

Table of Contents

Exercise 1: Populate family lookup field 2

Exercise 2: Populate family lookup field (Fixed) 6

Exercise 3: @invocableMethods with Complex data structures 10

Exercise 4: Display contacts grouped by families 16

Exercise 5: LWC As Custom Property Editors 22

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

Instructions:

1. First, let's see if this works
   - Open the **EX01 - Link family** flow
   - Click **Debug**
   - Select **Skip start condition requirements**
   - Select **Updated**
   - Type: **Perez**
   - Select any contact
   - Change Last Name to **Smith**
   - Click **Run**

   a. Which path did the flow go through?

   Family found

   b. Are the new records permanently stored in the database?

   No

   c. Why are the records not permanently in the database?

   Rolled-back

   - Click **Edit Flow** and **Activate** the flow

2. Review the **Start** element
   - Go back to the **EX01 - Link family** flow
   - Review the **Start** element
   a. What type of flow is this?

   Pirate (Action and Related Records)

   b. What object is this record-trigger flow firing on?

   Contact

3. Review the **Get010** element
   a. What is the equivalent SOQL query?

   SELECT *
   FROM Family__c
   WHERE LastName__c = :$Record.LastName
   ORDER BY N/A
   LIMIT 1

   b. Where do the results get stored?

   {!Get010}

4. Review the **IF020** element
   a. What is this condition validating?

   If a family record exists

5. Review the **Insert030** element
    a. Which record does it create?

    The missing family record

    b. Where do the results get stored?

    {!Get010.Id}

6. Review the **Update40** element
    a. Which record does it update?

    Original contact

    b. Which field is being updated?

    Lookup field to Family__c

7. Does it **really** work?
    - Open **HOW/main/default/classes/EX01_LinkFamily_Test.cls**
    a. What does the **testSetup()** method do in this file?

    Creates 2 contacts named LastName01 and LastName02

    b. How many contact records should be there **during the test execution**?

    2

    c. How many Family__c records should be there **during the test execution**?

    2

- Other methods will create more Contacts with different last name scenarios
- In line #1.5 click **Run All Tests**

d. Did all the tests succeed?

Yes

- In line #25, remove the **comment**
- **Save** and **deploy**
- In line #23.5 click **Run Test**

e. Did the tests succeed?

No

f. What error did we get?

Duplicate error

8. Deactivate flow
    - Go back to the flow and **deactivate** it!

Exercise 2: Populate family lookup field (Fixed)

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

Instructions:

1. First, let's see if this works
   - Open the **EX02 - Link family (Apex)** flow
   - Activate this flow
   - Open **HOW/main/default/classes/EX01_LinkFamily_Test.cls**
   - In line #1.5 click **Run All Tests**

   a. Did all the tests succeed?

   Yes

   b. Did we run the test on line 25? (remember it was failing before)

   Yes

   c. Was the test on line 25 successful?

   Yes

2. Review the **Apex010** element
   - 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}

   b. How many Contact records does **$Record** hold?

   One

- Expand the **Advanced** section

c. What is the name of the variable receiving the results from the Apex call?

{!contactWithFamily}

d. What is the data type of **contactWithFamily**?

Record (Single) | Contact

e. How many Contact records can **contactWithFamily** hold?

One

3. Review the **Update020** element

a. What is the variable name that holds the information we are updating?

{!contactWithFamily}

b. How many Contact records can **contactWithFamily** hold?

One

4. Review the **EX02_LinkFamily.cls** Apex class

- Open **HOW/main/default/classes/EX02_LinkFamily.cls**

a. In line #2, which annotation are we using?

@InvocableMethod

b. How many methods annotated with **@InvocableMethod** are in this class?

One

c. In line #3, how many parameters take the method?

One

d. What's the data type of the parameter **contacts**?

List<Contact>

e. How many Contact records can this variable hold?

Many

f. What's the return type?

List<Contact>

g. How many Contact records could be returned?

Many

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

- Bulkification
- Flow Transaction vs. Flow Interview

i. In line #25, which records are we inserting in that DML operation?

Missing families

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

None, DML is done in 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?

EX02_LinkFamily

Exercise 3: @invocableMethods with Complex data structures

Goal: Review how complex data structures work with @invocableMethods

Instructions:

1. First, let's see if this works
   - Open the **EX03 - DataStructures** flow
   - Click **View Tests**
   - In the list, click on the dropdown and select **Edit**
   - Click on the **Set Assertions** link

   a. On the first assertion, which property of results contains the LastName being compared with "Pérez"?

   - current
   - {!results.current.LastName}

   - Click **cancel**
   - In the list, click on the drop-down and select **Run Test and View Details**

   b. Did both assertions pass?

   Yes

2. Review the **Set010** element
   - Go back to the **EX03_DataStructures** flow
   - Review the **Set010** element

   a. How many Contact records can **$Record__Prior** hold? (Assume we are updating)

   One

   b. How many Contact records can **$Record** hold?

   One

c. What is the data type of the variable **contacts**?

Record (Collection) | Contact

d. How many Contact records can **contacts** hold?

Many

e. What are we doing in this assignment element?

Appending Prior and current value to contacts list

f. How many **pairs** of Contact records can **contacts** hold?

Many

g. We don't have a loop! How are we adding multiple pairs? (tricky question)

Flow transaction vs. flow interview

3. Review the **Apex020** element
   a. How many input values are we sending?

   2

   b. Is the first value (**Input Contacts**) required?

   Yes

c. What is the variable name we are sending?

{!contacts}

d. Is the second value (**The answer**) required?

No

e. What value are we passing in the second value (**The answer**) ?

42

f. Is that the right answer? (Geeky joke)

Yes

- Expand the **Advanced** section

g. What is the variable name receiving the result from the Apex call?

{!results}

h. What is the data type of **results**?

EX03_DataStructuresAuraEnabled

i. Does this variable accept one or many elements?

One

j. Why just one?

One per flow interview

4. Review the **EX03_DataStructures.cls** Apex class
   - Open **HOW/main/default/classes/EX03_DataStructures.cls**

a. In line #2, which annotation are we using?

@InvocableMethod

b. What's the data type of the parameter **requests**?

List<Request>

c. What determines the number of elements in **requests** parameter?

* The number of Flow Interviews per Flow Transaction
* The number of records in the DML for Record Triggered flows

d. Where (which line number) are we defining the **Request** class?

13

e. How many class variables are defined in the **Request** class?

2

f. Which annotation are we using for them?

@InvocableVariable

g. In line #7, what do the **requests[0]** mean?

The first flow interview

h. What do the requests[0]**.inputContacts[0]** mean?

The first contact (for the first interview)

i. In line #3, what's the return type for the @invocableMethod?

List<Result>

j. What determines the number of elements returned?

Same number of elements input

k. Where (which line number) are we defining the **Result** class?

20

l. How many class variables are defined in the **Result** class?

1

m. Which annotation are we using for **outputData**?

@InvocableVariable

n. What's the data type of **outputData**?

EX03_DataStructuresAuraEnabled

5. Review the **EX03_DataStructuresAuraEnabled.cls** Apex class
   - Open **HOW/main/default/classes/EX03_DataStructuresAuraEnabled.cls**

   a. How many properties are in the **EX03_DataStructuresAuraEnabled** class?

   2

   b. Which annotation are we using for them?

   @AuraEnabled

   - In question 1a, We saw the assertions used results**.current**.
     - 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

Instructions:

1. First, let's see if this works
   - Navigate to the Family application
   - Open the home tab
   - Click on the **Finish** button

   a. Which error did we get?

   You can't go past this screen!

   - Find the **Pérez** family

   b. How many Contact records are associated with this last name?

   31

   - Expand the **Pérez** family
   - Click on **Andrés**

   c. Did we navigate to the **Andrés Pérez** Contact record?

   Yes

2. Review the **Get010** element
   - Open the **EX04 - TreeGrid** flow
   - Review the **Get010** element

   a. What is the equivalent SOQL query?

   SELECT *
   FROM Family__c
   WHERE N/A
   ORDER BY N/A
   LIMIT N/A

   b. Where do the results get stored?

   {!Get010}

3. Review the **Apex020** element

   a. How many values are we sending?

   3

   b. What is **Families Input**?

   {!Get010}

   c. What else are we sending?

   Two more numbers, each 100.

   d. Where do the results get stored?

   {!Apex020}

4. Review the **EX04_QueryFamilies.cls** Apex class
   - Open **HOW/main/default/classes/EX04_QueryFamilies.cls**

   a. In line #12, we have an assertion. What does it validate?

   That we only have one interview

   b. Why can we make sure this assertion always passes?

   Because it's a screen flow we only have one interview

   c. In line #15 which Family__c records do we query?

   The same provided in the parameter

   d. Is there a maximum number of Family__c records we retrieve?

   Yes

   e. In line #15 which Contact records do we query?

   The ones related to each Family__c

   f. Is there a maximum number of Contact records we retrieve per Family__c?

   Yes

   g. In line #3, what is the return type of the method?

   List<FamilyResponse>

h. Which line number defines the **FamilyResponse** class?

43

i. What is the name of the class variable defined in the FamilyResponse class?

response

5. Review the **Screen030** element
   - Go back to the **EX04 - TreeGrid** flow and review the **Screen030** element

   a. How many custom components do we have in this flow?

   2

   b. What is the name of the custom component we are using?

   Family Tree Grid

   - Select the **Family Tree Grid** component

   c. What is the data we are using in **Family Types**?

   {!Apex020.response}

   d. Why can we use Apex020.**response**?

   @AuraEnabled

6. 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

   - Open the ex04TreeGrid.**js-meta.xml** file

   b. In line #5, what is the **masterLabel**?

   Family Tree Grid

   c. In line #7, what is the **target**?

   lightning__FlowScreen

   d. In line #11, what is the value of the property's **label** attribute?

   Family Types

   e. In line #11, what is the value of the property's **type** attribute?

   apex://EX04_FamilyTypeCollectionAuraEnabled

   f. In line #11, what is the value of the property's **name** attribute?

   familyTypes

   - Open the ex04TreeGrid.**js** file

   g. Are we using a **@wire**?

   No

h. How do we get the data without a @wire?

Setting familyTypes (@api)

i. In line #57, what's the error message?

You can't go past this screen!

j. In line #67, what are we doing?

Navigating to the Contact record

Exercise 5: LWC As Custom Property Editors

Goal: Use LWCs to configure the admin view of the flows.

Instructions:

1. First, let's see if this works
   - Open the **EX05 - Custom Properties** flow
   - New path has been added, the old path is the same as Exercise 4
   - Open the **Apex020 (New)** element

   a. Which standard LWC do we use to input **Families Data Provider**?

   lightning-combobox

   b. What are the options available?

   "None" and "Get010"

   - Select **None** and click **Done**

   c. What error did you get?

   Family is a required field. Select a value

   - Switch back to Get010

   d. Which standard LWC do we use to input **Max Families**?

   lightning-slider

   e. Which standard LWC do we use to input **Max Contacts Per Family**?

   lightning-slider

- Open the **Apex020 (Old)** element

f. Are the input widgets different?

Yes

- Open the **Screen030 (New)** element
- Click on **Family Tree Grid (EX05)**

g. Which standard LWC do we use to input **Data**?

lightning-combobox

h. How many options are available?

3

- Select **None** and click **Done**

i. Did you get an error?

No

- Open the **Screen030 (New)** element
- Switch back to **Apex020 (New) Get families with contacts**
- Click **Done**
- Open the **Screen030 (Old)** element
- Click on **Family Tree Grid**

j. Is the input widget for **Family Types** different?

Yes

2. Review the **EX05_QueryFamilies.cls** Apex class
   - Open **HOW/main/default/classes/EX05_QueryFamilies.cls**
   - This class is identical to the previous one except for one difference...
   a. In line #6, which attribute of the @invocableMethod are we configuring?

      configurationEditor

   b. In line #6, what's the value for **configurationEditor**?

      c-ex05-apex-configuration

3. Review the **ex05ApexConfiguration** LWC component
   - Find the **HOW/main/default/lwc/ex05ApexConfiguration** folder
   - Open the ex05ApexConfiguration.**html** file
   a. In line #5, which LWC do we use to input **Families Data Provider**?

      lightning-combobox

   b. In line #20, which LWC do we use to input **Max Families**?

      lightning-slider

   c. In line #31, which LWC do we use to input **Max Contacts Per Family**?

      lightning-slider

   - Open the ex05ApexConfiguration.**js** file
   d. In line #26, which setter do you find?

      inputVariables

e. In line #45, which setter do you find?

builderContext

f. In line #49, how do we find the list of **Get** elements in the flow?

recordLookups

g. In line #65, what is the error message?

Family is a required field. Select a value

h. In line #99, how does LWC notify the flow that we selected a different value?

dispatching a custom event named "configuration_editor_input_value_changed"

4. Review the **ex05TreeGrid** LWC component
   - Find the **HOW/main/default/lwc/ex05TreeGrid** folder
   - Open the ex05TreeGrid.**js-meta.xml** file

   a. In line #10, which **configurationEditor** are we specifying?

   ex05LwcConfiguration

   - Find the **HOW/main/default/lwc/ex05LwcConfiguration** folder
   - Open the ex05LwcConfiguration.**html** file

   b. In line #5, which LWC do we use to input **Data**?

   lightning-combobox

- Open the ex05LwcConfiguration.**js** file

c. In line #16, which setter do you find?

inputVariables

d. In line #34, which setter do you find?

builderContext

e. In line #39, how do we find the list of **Apex Action** elements in the flow?

actionCalls

f. In line #56, how does LWC notify the flow that we selected a different value?

dispatching a custom event named "configuration_editor_input_value_changed"