dreamforce

Enhance Flows with Apex and Lightning Web Components

Flow + Apex + LWC = A Perfect Combination!

Dreamforce 2024

Wed Session (#1)

Andrés Pérez
Salesforce
Senior Manager, Solution Architect Lead

Joseph Flowers
Salesforce
Senior Mgr, Learning Solutions Architect

Thur Session (#2)

Rahul Shah
Salesforce
Manager, APAC Delivery

Tripty Jha
Salesforce
Sr. Technical Instructor

Forward looking statements

This presentation contains forward-looking statements about, among other things, trend analyses and statements regarding future events, anticipated growth and industry prospects, and our strategies, expectation or plans regarding product releases and enhancements. The achievement or success of the matters covered by such forward-looking statements involves risks, uncertainties and assumptions. If any such risks or uncertainties materialize or if any of the assumptions prove incorrect, results or outcomes could differ materially from those expressed or implied by these forward-looking statements. The risks and uncertainties referred to above include those factors discussed in Salesforce's reports filed from time to time with the Securities and Exchange Commission, including, but not limited to: our ability to meet the expectations of our customers; uncertainties regarding AI technologies and its integration into our product offerings; the effect of evolving domestic and foreign government regulations; regulatory developments and regulatory investigations involving us or affecting our industry; our ability to successfully introduce new services and product features; our ability to execute our business plans; the pace of change and innovation in enterprise cloud computing services; and our ability to maintain and enhance our brands.

Last updated: April 25, 2024

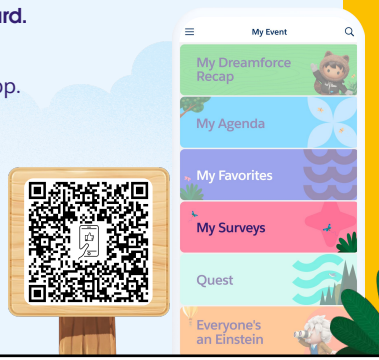
Copyright

Coffee on us!

The first 4,000 attendees to provide feedback on this event will receive a $5 Starbucks gift card.

1. Open the Salesforce Events mobile app.
2. Navigate to **My Event**.
3. Select **My Surveys**.
4. Complete four session surveys and present the completed Event Survey page at Badge Pickup to redeem.*

*Restrictions apply. See rules at sforce.co/survey-terms



Agenda

01 Using record-triggered flows

02 Using Apex in flows

03 Combining lightning web components (LWCs) and flows

04 Wrap and Q&A

Setup: Set up your environment for this session

Instructions Summary:

1. Log in to the online session site.
2. Log in to your training session Salesforce org with the username and password provided to you.
   - URL: `login.salesforce.com`
   - Username: Replace `###` with your laptop's number:

Wed. Session (#1): `df24@apexflows-A.###.com`

Thur. Session (#2): `df24@apexflows-B.###.com`

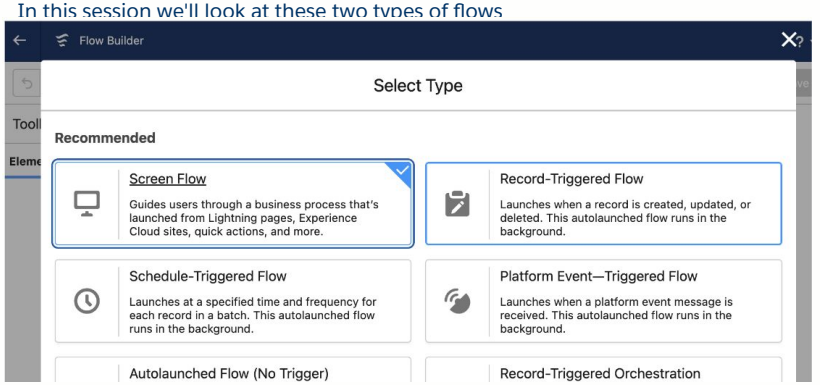
3. Enable and launch Code Builder.

Automating business processing by using flows

Code completion enhanced!

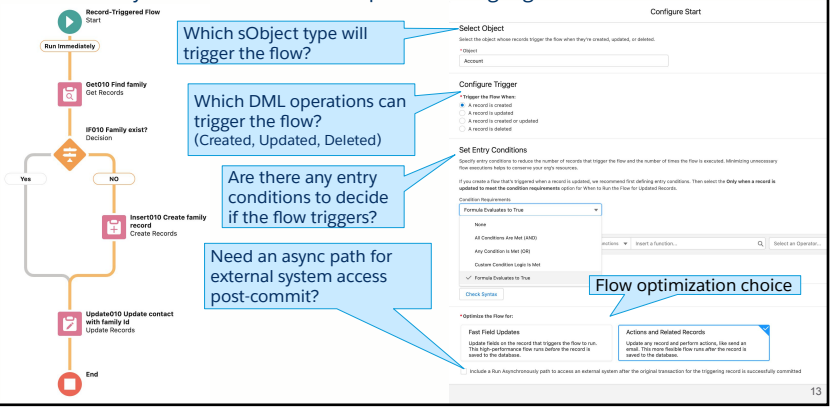
Types of flows

In this session we'll look at these two types of flows

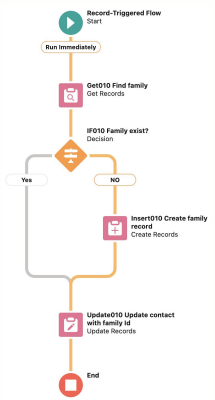


Record-triggered flows run in the background (no UI)

Automatically executed on Data Manipulation Language (DML) events



Deciding between flow optimization options



Fast Field Updates

- Update fields on the record that **triggers the flow** to run
- This **higher-performance** flow runs ***before*** the record is saved to the database

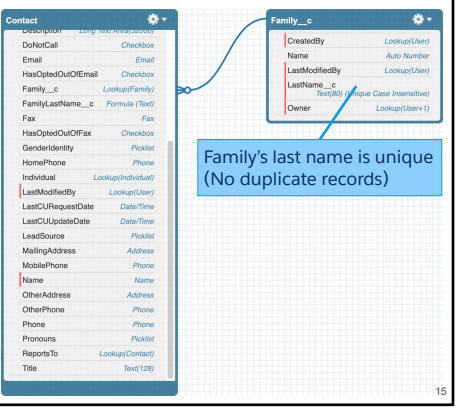
Actions and Related Records

- Update **any** record and perform actions, like sending an email
- This **more flexible** flow runs ***after*** the record is saved to the database

Linking a contact with a family record

Business requirement for the exercise

- **Contacts** should be related to their **family** records, based on their **last name**
- Whenever a **contact** is **created** or **updated**, find a related **family** record and **link** them together
- If a **family** record is not found, a new family record should be **created** and **linked**



Consider this before we start the exercises:

~~Apex developers don't write code!~~

We invest more time in reading code than in writing it!

- Code reviews
- Troubleshooting / Debugging
- Einstein for Developers

In the exercises, you will read and analyze pre-built code examples

Exercise 1: Analyze and test a Salesforce flow

Goals:

Explore limits of using flows to automatically insert related family records when a collection of contact record change.

Instructions Summary:

1. Debug a pre-built Salesforce flow.
2. Analyze what the flow does.
3. Test behavior of an activated vs. deactivated flow.
4. Test the flow.
5. Identify issues with and limitations of the flow.

Why did the test fail?

The error said that duplicate records cannot be inserted

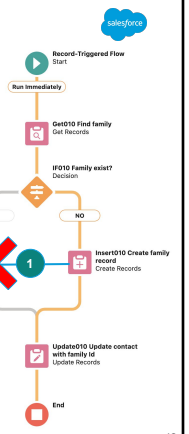
```
OUTPUT  ···   Apex
EX01_LinkFamily_Test.
DmlException: Insert failed. First exception on row 1; first
error: CANNOT_EXECUTE_FLOW_TRIGGER, We can't save this
record because the "EX01 – Link family" process failed. Give
your Salesforce admin these details. This error occurred
when the flow tried to create records: DUPLICATE_VALUE:
duplicate value found: <unknown> duplicates value on record
```

DML Operation

1 2

Contact records

1	Smith
2	Smith



Working separate but together!

Flow interviews vs. Flow transactions

DML operation

1 2

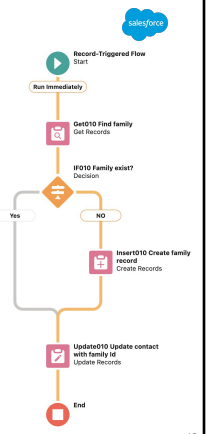
1	Smith
2	Smith

Triggers

Flow transaction

Flow interview 2

Flow interview 1



It works!

With **different** new last names

- **Bulkification** points
  - **SOQL / DML operations**
  - Screen elements
  - Pause actions
  - Apex calls (optional)

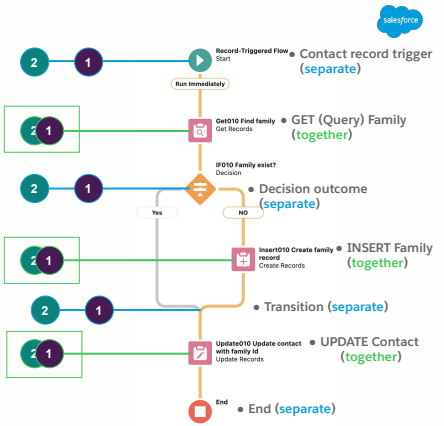
DML Operation

1 2

Contact records

1 Pérez

2 Smith



It fails!

With the **same** new last names

- Inserts duplicate records in the flow transaction (bulkified operation)

DML Operation

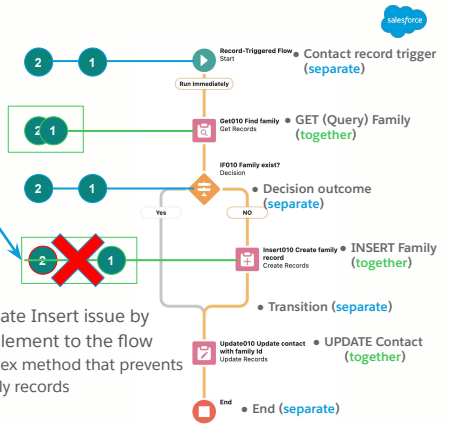
1 2

Contact records

1 Smith

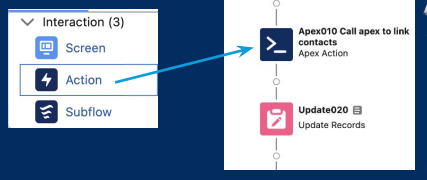
2 Smith

- Next, we'll fix this duplicate Insert issue by adding an Apex action element to the flow
  - This element calls an Apex method that prevents inserting duplicate Family records



Customize flows by adding Apex actions

Why and how to use Apex in flows



Record-triggered flows vs. Apex triggers

- **Flows** let you create customized actions visually in Flow Builder **without writing code**
- **Apex** lets you create customized actions **with code**, usually in an IDE like Code Builder
  - Recommended to use flows as much as possible

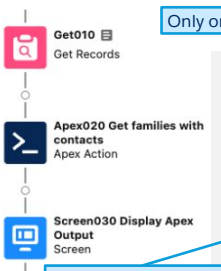

```
trigger phoneFieldTrg on Account (before Insert)
{
  if(trigger.isBefore && trigger.isInsert)
  {
    if(!trigger.new.isEmpty())
    {
      for(Account acc : trigger.new)
      {
        if(acc.Phone == null || acc.Phone == '')
        {
          acc.addError('You cannot insert account
with phone field empty');
        }
      }
    }
  }
}
```

Answers: Comparing record-triggered flows vs. Apex triggers

	Record-triggered flows	Apex triggers
Stops DML transaction (validations)	Yes	Yes
Entry condition	Automatic	Manual
Asynchronous path	Automatic	Manual
Undelete event	No	Yes
How many (per object/event)?	Many	One (Best Practice)
Queries	Simple	Relationships, Aggregates
Bulkification	Automatic	Manual
New record	`$Record` (1 record)	`Trigger.new` (list)
Old record	`$Record__Prior` (1 record)	`Trigger.old` (list)

Combine flows and Apex

Invoke Apex methods from flows by using Action elements



Only one per class

Class access with permission sets (or profiles)

```
1  public without sharing class RT_LinkFamily {
2    @InvocableMethod(label='Link Families'
3        description='...'
4        category='Contact' [Other modifiers]
5    )
6    public static List<Contact> linkFamily(
7      List<Contact> contacts ) {
8      ...
9    }
10 }
```

Developer must **bulkify** the method

Only one input parameter and it must be a List

Result must be a List with same size (or null), and records must be in the same order as input

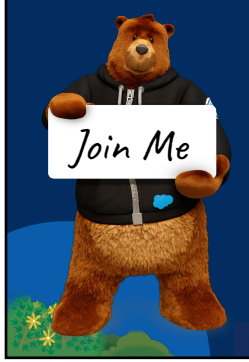
Solution: Can we use flows or do we need Apex? (or both)

No code → Pro code

	F.F.U. flow	A.R.R. flow	A.R.R. flow + Apex Class	Apex Trigger / Classes
Same-Record Field Updates	Available (Best)	Available (Not Ideal)	Available (Not Ideal)	Available (OK)
CRUD operations	Not Available	Available (Best)	Available (OK)	Available (OK)
Asynchronous Processing	Not Available	Available (Best)	Available (OK)	Available (OK)
Complex List Processing with Map/Set	Not Available	Not Available	Available (Best)	Available (OK)
High-Performance Batch Processing	Available (Not Ideal)	Available (Not Ideal)	Available (Not Ideal)	Available (Best)
Complex SOQL Queries	Not Available	Not Available	Not Available	Available (Best)
Custom Validation Errors	Available (Best)	Available (OK)	Available (OK)	Available (OK)

https://architect.salesforce.com/decision-guides/trigger-automation

Exercise 2: Analyze how to fix issues and limitations in a flow by using an Apex action element



Goals:

Explore how an APEX invocable method is used to customize a flow to link Contact and Family records.

Instructions Summary:

1. Import the starter app from GitHub to Code Builder.
2. Authorize Code Builder with your training session Salesforce org.
3. Run Apex tests for a flow in Code Builder.
4. Analyze how Apex code is connected with the flow.
5. Analyze the Apex code used to customize the flow.
6. Analyze how the @invocableMethod annotation is used.
7. Analyze the permission set used with the flow.

Exposing Apex variables to flows

Customize the user experience and perform more complex actions

Share Apex variables in custom data classes by using the @invocableVariable annotation

Each custom type can contain multiple invocable variables

The label appears in Flow Builder as info to help admins use the variable

Custom Apex-defined class

```
public with sharing class RT_Demo {
  @InvocableMethod(label='Sample' description='...' category='Contact')
  public static List<Result> execute(List<Request> requests) {
    List<Result> responseWrapper = new List<Result>();
...
  }
  //Inner classes - Custom data types
  public class Request {
    public String notAccessible;

    @InvocableVariable(label='Input Contacts' required='true')
    public List<Contact> inputContact;

    @InvocableVariable(label='Family of all Contacts' required='false')
    public Family theFamily;
  }

  public class Result {
    @InvocableVariable(label='The Ultimate Answer' required='true')
    public String theAnswer;

    @InvocableVariable(label='Output Contact' required='true')
    public RT_DemoAE outputContact;

  }
}
```

Making custom data attributes available to flows by using the @AuraEnabled annotation

```apex
public with sharing class RT_Demo {
  @InvocableMethod(label='Sample' description='...' category='Contact')
  public static List<Result> execute(List<Request> requests) {
    List<Result> responseWrapper = new List<Result>();
    Result response = new Result();
    response.outputContact = new RT_DemoAE();
    response.outputContact.myContact = requests[0].inputContact[0];
  }
  public class Request {
    @InvocableVariable(label='Input Contacts' required='true')
    public List<Contact> inputContact;
  }
  public class Result {
    @InvocableVariable(label='Output Contact' required='true')
    public RT_DemoAE outputContact;
  }
}
```

- @AruaEnabled makes the property accessible in flows
- Serializes the property

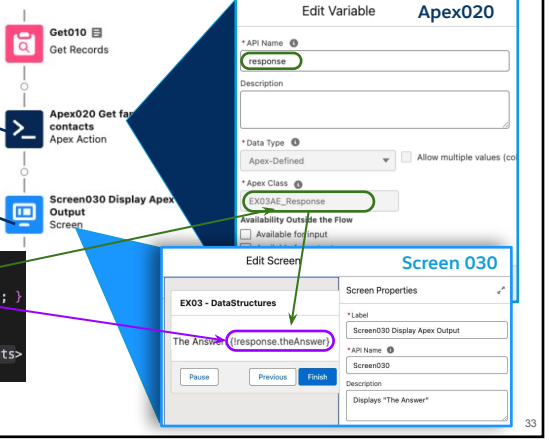
- Custom data type for variable
- Must be in a separate outer class file

```apex
public class RT_DemoAE {
    @AuraEnabled
    public Contact myContact { get; set; }
    @AuraEnabled
    public Integer theAnswer { get; set; }
}
```


Example of Flow elements accessing @AuraEnabled attributes

1. External **Apex** data class is used in an invocable method response.

2. The next flow element can access the **@AuraEnabled** variable.




```apex
public class EX03AE_Response {
    @AuraEnabled
    public Integer theAnswer { get; set; }

    @AuraEnabled
    public List<EX03AE_FamilyWithContacts>
    familiesWithContacts { get; set; }
```

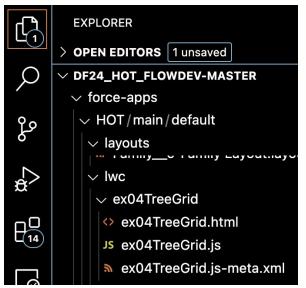
Combining Lightning web components (LWCs) and screen flows

Building reusable components of customized screens

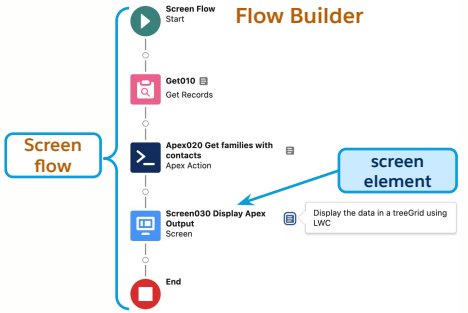
Comparing how LWCs and screen flows are built

- LWCs are built in a standard web development tool (IDE) such as Salesforce Code Builder
  - **Pro-code**
- Screen flows are built in Flow Builder
  - **Low-code / no-code**

LWC in an IDE



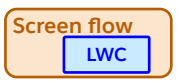
Flow Builder



Ways to use screen flows and LWCs together

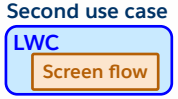
Two opposite use cases to reuse screen flows and LWCs

First use case



- Screen flow UI component embeds (display) LWC element(s) inside
  - To customize part of a screen element user experience (UX) by reusing an LWC

Second use case

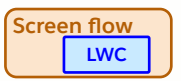


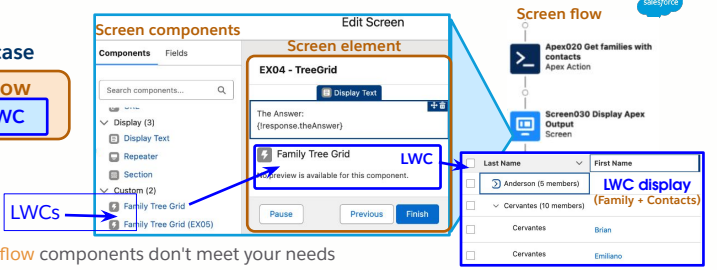
- The LWC embeds (displays) a screen flow inside
  - To reuse screen flow components that can be built with no-code
  - To restrict access to a screen flow to specific sections of the Lightning Experience (via the target property of the LWC)

- You can even combine both of these patterns in your UI
  - Leverage the best of no-code and pro-code options
  - Promote reusability

Why embed an LWC in a screen flow?

First use case





- If OOTB flow components don't meet your needs
  - Add LWC screen components as part of a screen's layout
  - Build reusable advanced UI options such as pixel perfect, 4+ columns, tabs, colors, buttons, sliders, custom CSS, ...
- To create interactive screens (onblur, onfocus, onchange) with events not available to a flow
  - On the client-side, perform extra validation not available OOTB in the flow
  - Dynamically control requiredness, formatting, visibility, and/or read-only parts of the UI

Example: How to embed an LWC in a screen flow

Screen flow
LWC

Make an LWC available to a screen flow

LWC component.js-meta.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3   <apiVersion>57.0</apiVersion>
4   <isExposed>true</isExposed>
5   <masterLabel>Best Component Ever</masterLabel>
6   <targets>
7     <target>lightning__FlowScreen</target>
8   </targets>
9   <targetConfigs>
10    <targetConfig targets="lightning__FlowScreen">
11      <property name="startDate" label="Start Date" type="Date" role="inputOnly"/>
12      <property name="name" label="Account Name" type="String" role="outputOnly"/>
13      <property name="account" label="Account" type="@salesforce/schema/Account[]"/>
14      <property name="data" label="Custom Data" type="apex://CustomClass"/>
15    </targetConfig>
16  </targetConfigs>
17 </LightningComponentBundle>
```

Specify the LWC is available for screen flows

Properties passed between screen flow and LWC

Complex data types (Apex)

How an LWC dispatches events to the screen flow

Screen flow
LWC

LWC code to dispatch screen flow events to the "parent" screen flow

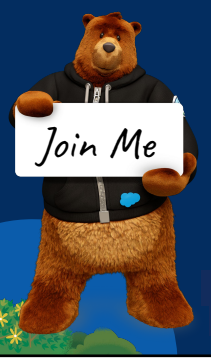
```
1 <!-- LWC component ES Module JavaScript .js file -->
2 import { LightningElement, api, track} from 'lwc';
3 import { FlowAttributeChangeEvent, FlowNavigationNextEvent } from 'lightning/flowSupport';
4
5 export default class LwcInFlow extends LightningElement {
6   @track _someData;
7   @api availableActions = [];
8   @api get someData() { return this._someData; }
9       set someData(value) { this._someData = {...someData}; }
10  ...
11  demo() {
12    this.dispatchEvent(new FlowAttributeChangeEvent('someData', this._someData));
13    ...
14    if (this.availableActions.find((action) => action === 'NEXT'))
15      this.dispatchEvent(new FlowNavigationNextEvent());
16  }
```

LWC notifies screen flow when values changed
("Reactive Screens")

Control screen flow navigation
Back, Next, Pause, Finish

Determine available screen flow actions
Example: If the LWC is the last screen component, don't dispatch Next event

Exercise 4: Analyze how to code and embed a Lightning Web Component (LWC) in a screen flow



Goals:

Explore how LWC are used with flows in the training session scenario to display Contact records grouped by the Family field.

Instructions Summary:

1. Analyze how the solution flow works.
2. Analyze how the solution was built in Flow Builder.
3. Analyze how the LWC was built in VS Code.

Summary: Flows + Apex + LWCs

Leverage and combine the best of no-code and pro-code options

- Flows provide a graphical (no-code) way to improve the UI and add logical structure to the user experience based on business requirements
  - Helps you to build UI screens and automated business processes without needing to build out your own infrastructure
- Apex and LWCs are helpful to enhance and customize flows to meet specific requirements not supported by OOTB flow components
  - Example: Using Maps or Sets, avoid DML bulkification issues, custom input/output objects
- Apex code can be leveraged directly from a flow to customize the behavior
- LWCs are customized with HTML and JavaScript code, and can be used inside flows

dreamforce

Thank you

dreamforce

Q&A

Coffee on us

The first 4,000 attendees to provide feedback on this event will receive a $5 Starbucks gift card.

1. Open the Salesforce Events mobile app.
2. Navigate to **My Event**.
3. Select **My Surveys**.
4. Complete four session surveys and present the completed Event Survey page at Badge Pickup to redeem.*

*Restrictions apply. See rules at sforce.co/survey-terms