Process Automation

Workflow Rules, Process Builder, Flow and Triggers

Joanna Iturbe, Leeds School of Business (University of Colorado)
Kyle Schmid, Huron Consulting Group

February 25, 2020

Agenda

- Crowd Pulse Check
- Which Tool Do I Use?
- Workflow Rules Review and Use Cases
- Process Builder Review and Use Cases
- Flow Review and Use Cases
- Apex Trigger Review and Use Cases
- Demo
- Audience Input and Q&A

Crowd Pulse Check

How many of you have used:

- Workflow Rules
- Process Builder
- Apex Triggers
- Flow

Need a volunteer for each to share a current use case:

- Workflow Rules
- Process Builder
- Apex Triggers
- Flow

For those using the tool(s), what is your pain point with each:

- Workflow Rules
- Process Builder
- Apex Triggers
- Flow

If you could learn to use one tool better, which one would you pick?

- Workflow Rules
- Process Builder
- Apex Triggers
- Flow

Which Automation Tool Do I Use?

	PROCESS BUILDER	FLOW BUILDER	WORKFLOW
Complexity	Multiple if/then statements	Complex	A single if/then statement
Visual designer	✔	✔	
Starts when	• Record is changed • Invoked by another process • Platform event message is received	• User clicks button or link • User accesses Lightning page, Community page, Visualforce page, or custom tab • User accesses item in a utility bar • Process starts • Apex is called	Record is changed
Supports time-based actions	✔	✔	✔
Supports user interaction		✔	

Source: Salesforce Help

Which Automation Tool Do I Use? *cont.*

	PROCESS BUILDER	FLOW BUILDER	WORKFLOW
		Supported Actions	
Call Apex code	✓	✓	
Create records	✓	✓	Tasks only
Invoke processes	✓		
Delete records		✓	
Launch a flow	✓	✓	✓ (Pilot)[1]
Post to Chatter	✓	✓	
Send email	✓ (Email alerts only)	✓	✓ (Email alerts only)
Send custom notification	✓	✓	
Send outbound messages without code			✓
Submit for approval	✓	✓	
Update fields	Any related record	Any record	The record or its parent

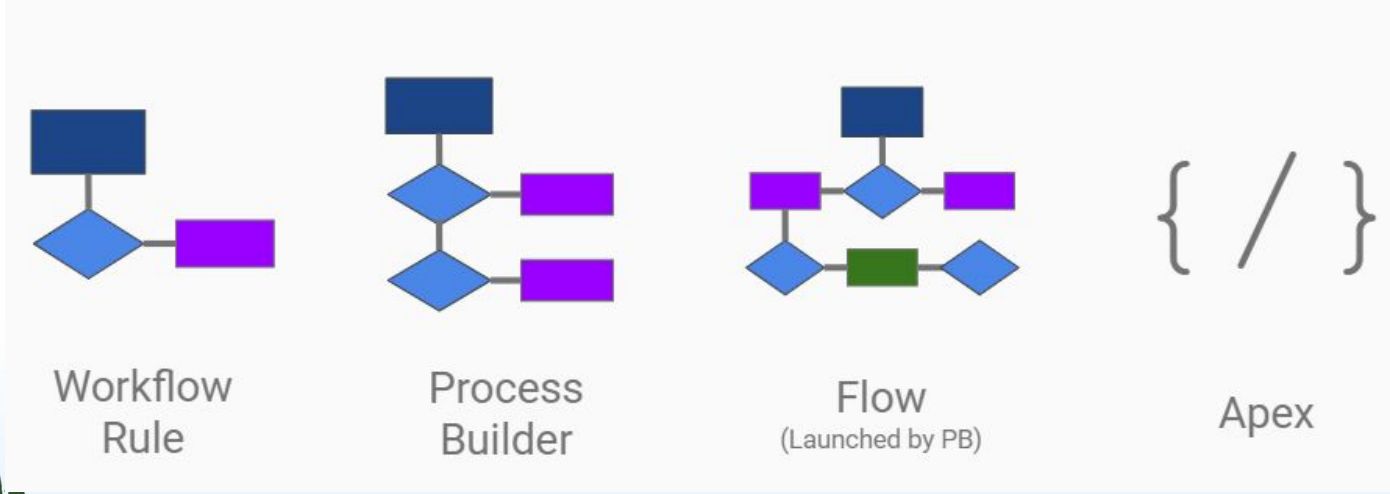
Source: Salesforce Help

Salesforce Golden Rule: Use the Simplest Tool for the Job!

(...Except when you shouldn't)

Lightning era has brought a lot of grey area to the 'Golden Rule'
Certain use cases value X over simplicity

- Bulk data load = Speed
- Actions on commonly-used objects = Low Risk
- Advising = User Experience

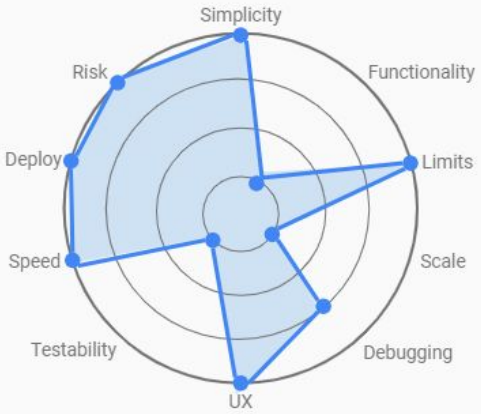


Source: David Liu - Tahoe Dreamin' 2018; Clicks vs. Code

Salesforce Process Automation Showdown

salesforce.org

Workflow Rule



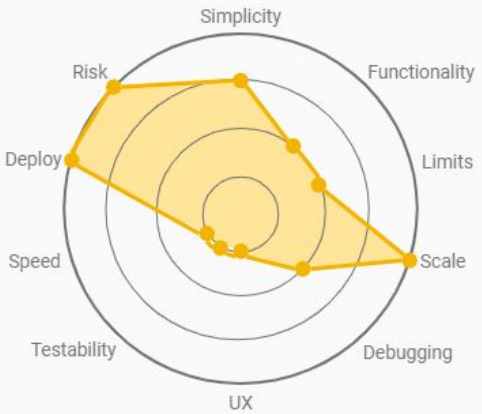
Pros:

- Simplicity
- Low Risk
- Easy to Deploy

Cons:

- Low Functionality
- Little Ability to Scale
- Low Testability

Process Builder

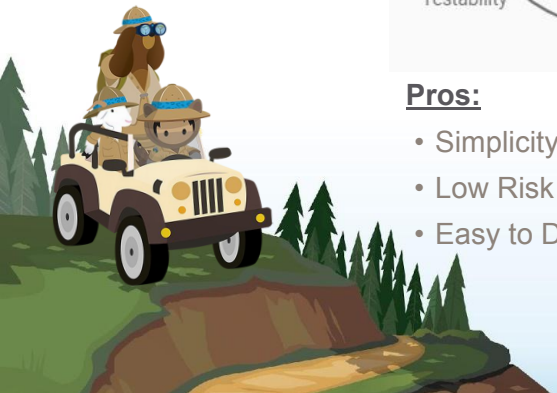


Pros:

- Low Risk
- Easy to Deploy
- Scalable

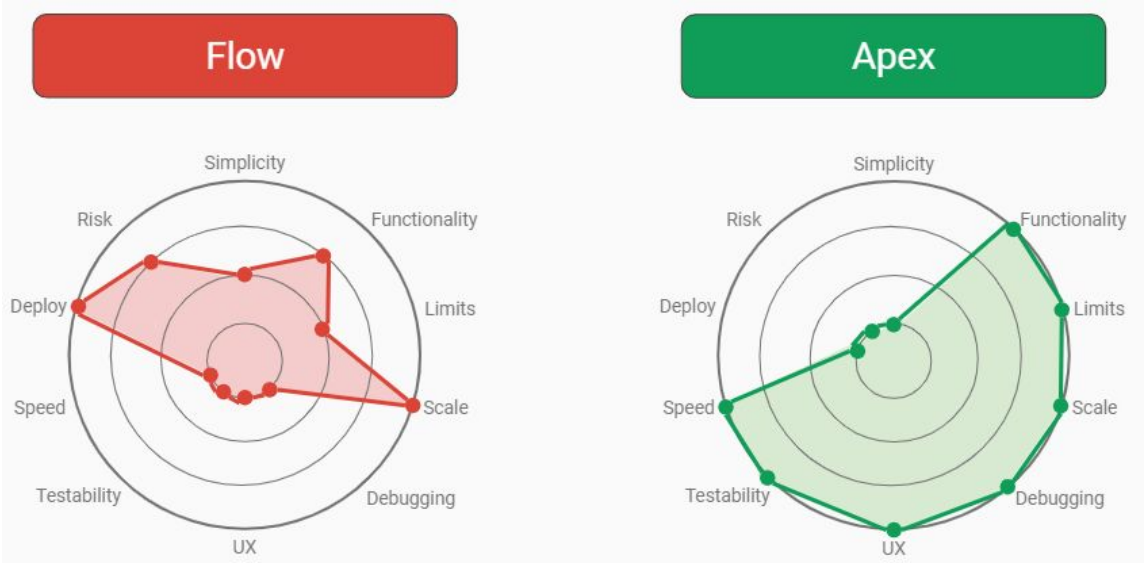
Cons:

- Bad UX
- Low Speed
- Low Testability



Source: David Liu - Tahoe Dreamin' 2018; Clicks vs. Code

Salesforce Process Automation Showdown *cont.*



Pros:

- Scalable
- Easy to Deploy

Cons:

- Ability to debug
- Speed
- Low Testability

Pros:

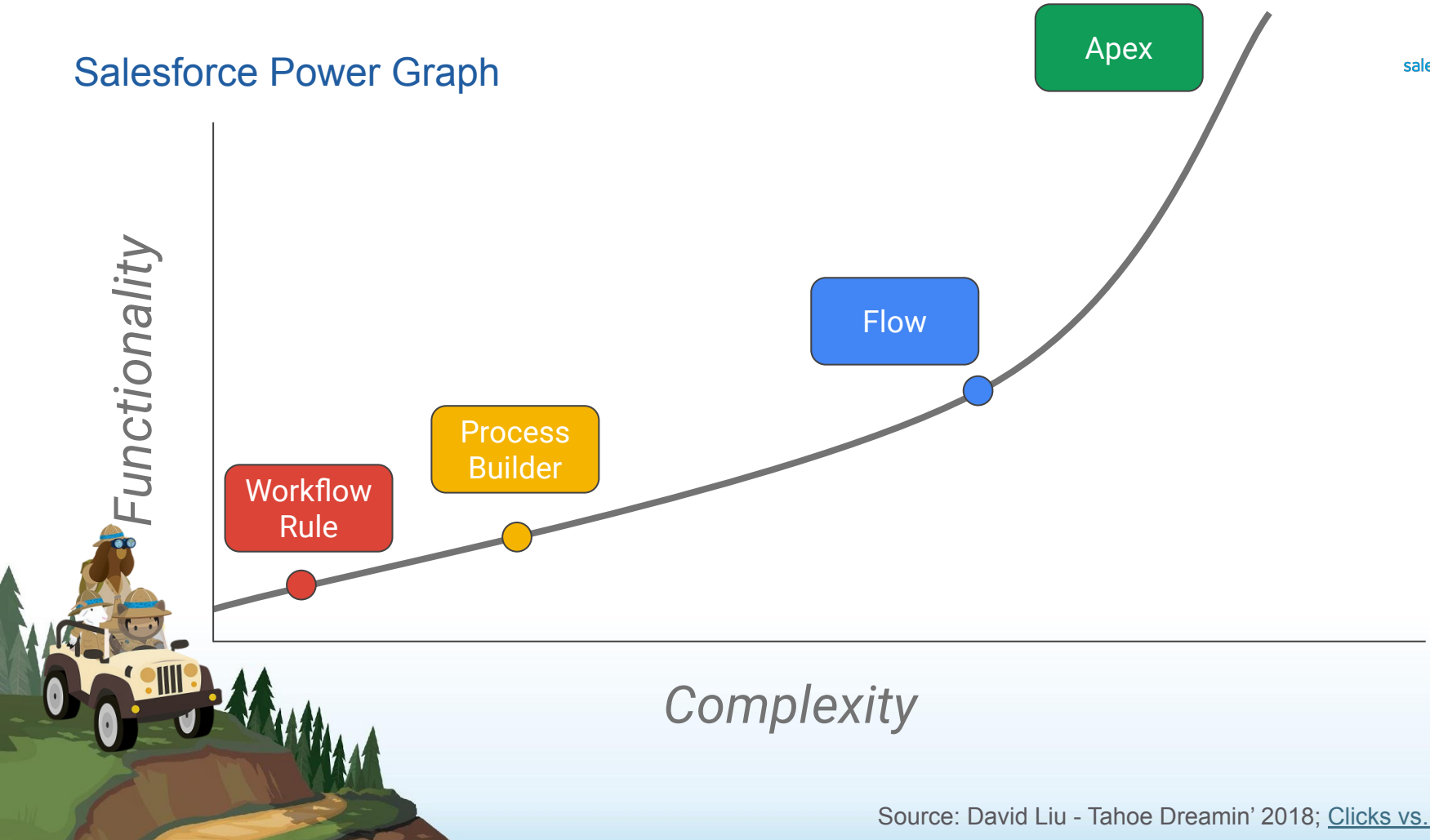
- Functionality
- Debugging
- Limits

Cons:

- Slower to Deploy
- High Risk
- Complex

Source: David Liu - Tahoe Dreamin' 2018; Clicks vs. Code

Salesforce Power Graph



Source: David Liu - Tahoe Dreamin' 2018; Clicks vs. Code

Are there any use cases left for workflow rules?

Survey says...

Yes!

- A single IFTT (if this, then that) statement to avoid recursion caused by Process Builder
- Outbound Messaging

Examples

- Remind me when it's a donor's birthday
- Assign a task to an advisor when a student's record has been inactive for certain period of time
- Send an email to a recruiter when a prospective student's opportunity score reaches a certain level
- Update a field in Salesforce when another field's value changes

Source: Funnel Fox Blog

Workflow Rule Migration

Trailhead Module

Salesforce Support

- No longer enhancing workflow rules
- Will support workflow rules for a TBD amount of time
- Recommend migrating to Process Builder (or leveraging Flow)

Easy Wins

- Any new objects -> Only use Process Builder / Flow
- Any new automation needed -> Only use Process Builder / Flow

Source: Salesforce Ben

Process Builder: Best Practices

10. Create reusable actions / processes

- Create common actions / processes / activities (i.e. email send action, update record action, etc.) to be reused by multiple processes
- Saves time and effort
- Done by using available Quick Actions or by creating these actions in an invocable process
  - Invocable processes: One that can be started only when initiated by another process

9. Avoid overlapping automation on one object

- Choose one type of automation per object (workflow rules, process builder, flow, triggers)
- Implement one process per object
- Makes it difficult to define order of operations and verify results
- Exceptions exist

8. Be aware of what can cause exceptions

- Example: Cross-object reference without checking for NULL

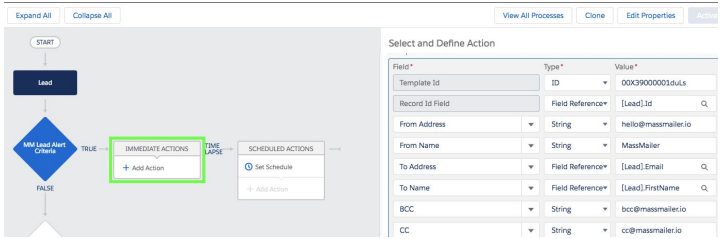
7. Look out for infinite loops

- Ex: A ‘create record’ action in Process 1 triggers Process 2 which contains an action of update record which is the trigger for Process 1
- Leads to breaching org limits

Source: Inspire Planner Blog

Best Practices: Process Builder *cont.*

6. Restrict the total number of actions in a process
   - Create fewer number of actions (nodes) and try to accommodate multiple requirements in a single action (node) to avoid reaching org limits like # of DML statements or total CPU usage
   - Ex: Update multiple fields in an object in one action rather than creating individual actions for individual fields



5. Look out for action timelines
   - An immediate action within a ‘node’ on a process may disqualify it for another scheduled action

4. Use scheduled actions to access external data

3. Deployment
   - Use a Sandbox
   - If possible, leverage Apex to test
   - Update hard-coded IDs after deployment (including hidden) - Ex: Post to Chatter

2. Follow a naming convention

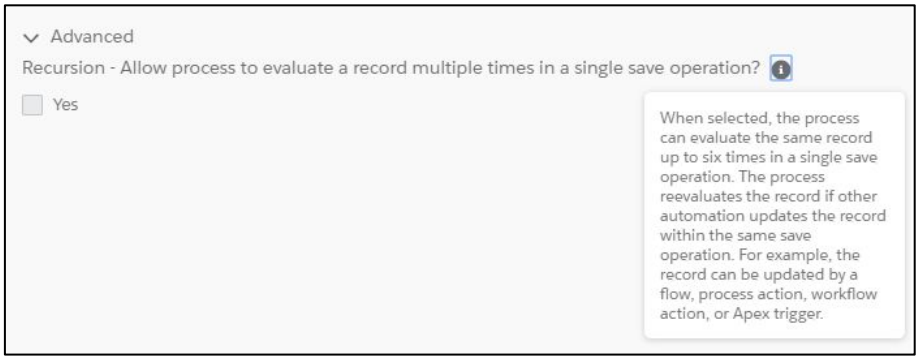
1. Use the Trailblazer community

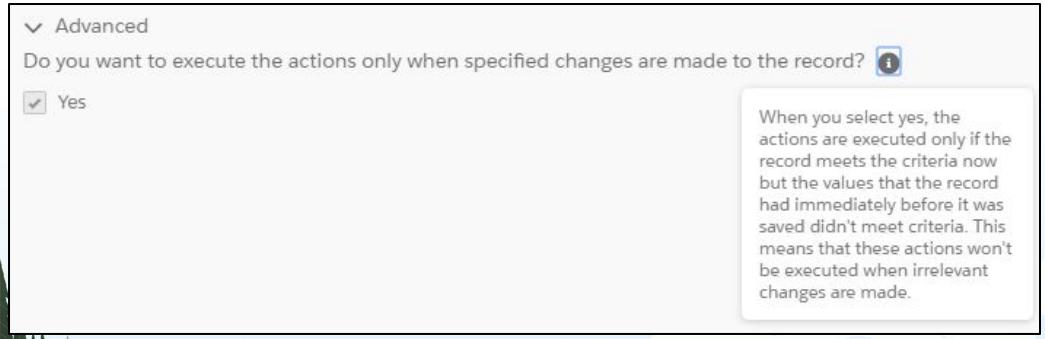
Source: Inspire Planner Blog

So many checkboxes, so little time...

Recursion checkbox

- Object-level
- Check if other automation may require this process to run more than once in a single transaction
- Ensure logic executes when 2+ updates on the same object occurs in same transaction
- Do not check if subsequent update doesn't require process logic to be re-evaluated





Specified Changes checkbox

- Criteria-level
- Check if not using an 'ISCHANGED' criteria
- Pair with criteria that determines whether or not logic needs to execute
- Ensure logic doesn't execute unnecessarily
- Can't be checked when using 'ISCHANGED' criteria

Flow

Use Cases

Autolaunched Flow

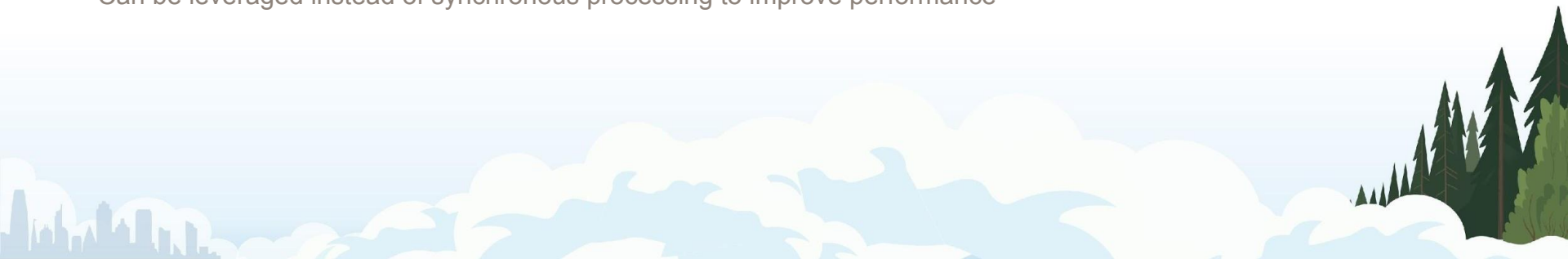
- Triggered by record updates
- Can populate fields on records in a before context (new in Spring ‘20)

Screen Flow

- Allows flow to display data and get input from users
- Can take different actions based on user input
- Can be used in Communities

Scheduled Flow

- Can perform scheduled operations on a batch of records based on filters you define
- Similar to Batch Apex
- Can be leveraged instead of synchronous processing to improve performance



Flow

Pros and Cons

Pros

- Coding concepts in a drag-and-drop interface
- Can “get” (query) records
- Can update records in a before-save context (new in Spring ‘20)
- Can update records on other objects
- Can perform complex conditional logic
- Supports versioning

Cons

- No way to write unit tests without using Apex
- Enforces security/permissions
  - Spring ‘20 allows Flows to ignore object/field-level permissions but not record-level permissions
- No good way to catch/log unhandled exceptions
- Doesn’t work well with version control tools

Flow

Best Practices

- Always check for null when accessing variables
- Consider all edge cases that could exist
- Run existing Apex tests in org before deploying
- If possible, write Apex tests to test your Flows
- Make sure end users have access to data/metadata used in Flows
- Login as end users to test Flows
- Leverage sub-flows to simplify development and re-use logic

Apex Triggers

Use Cases

Integration

- Can perform web service callouts

Delete Processing

- Can run before/after delete

Performance

- If written correctly (ahem), will provide peak performance

Asynchronous Processing

- Can call @future methods to process records asynchronously

Generic/Dynamic Logic

- Apex can be written that applies to generic SObjects/fields that can be re-used across objects

Other

- Access to all Apex libraries that unlock the full potential of Salesforce

Apex Triggers

Pros and Cons

Pros

- Can run in before and after context for insert, update, delete, and undelete
- Access to all classes in Apex library
- Can run in a system context
- Requires Apex tests
- Can kick-off asynchronous processes

Cons

- Cannot control order of execution among relative to other triggers
- Requires a developer
- Subject to Apex governor limits

Apex Triggers

Best Practices

- One trigger per object
- Separate logic into different Apex classes
- Iterate over lists/collections
- Use a trigger handling framework (EDA comes with TDTM)
- Filter records before processing
- Write Apex tests to test trigger logic, not to get code coverage
- Follow general Apex best practices

DEMO

Additional Resources

General

- Business Process Mapping for Salesforce Admins (30-minute webinar)
- Which Automation Tool Do I Use? (Salesforce Help Article)
- Workflow, Process Builder, Flow, or Apex? (Article and Presentation)
- Salesforce Virtual Workshops (Free)

Workflow Rules

- Workflow Rule Migration (Trailhead)

Flow

- Introduction to Flow (30-minute webinar)
- Lightning Flow (Trailhead)
- Be an Innovator with Flow Builder (Trailhead)
- New Flow Features: Winter ‘20 Spring ‘20

Triggers

- Triggers and Order of Execution (Help Article - Includes all automation types)


thank you

salesforce.org

Title Slide A (Standard)
Subtitle

Presenter Name, Title of Presenter
email@salesforce.com, @twitterhandle

Basic Layout – Title Only

Subtitle placeholder

Basic Layout

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

2 Column

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

3 Column

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Basic Mountain Footer

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Above the Trees

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Foothills - Left

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Scene Left - Mountain

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Scene Right - Lookout

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Scene Left - View

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Scene Right – Mountain Peak

Subtitle placeholder

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

First level text is set at twenty point Salesforce Sans and does not have a bullet

- Second level is eighteen point Salesforce Sans

Segue B

Subtitle

Segue C

Subtitle