Exam.4 Triggers and Order of Execution

2018年7月11日 18:06

On the server, Salesforce:

1. Loads the original record from the database or initializes the record for an upsert statement.
2. Loads the new record field values from the request and overwrites the old values.
   If the request came from a standard UI edit page, Salesforce runs system validation to check the record for:

- Compliance with layout-specific rules
- Required values at the layout level and field-definition level
- Valid field formats
- Maximum field length

   When the request comes from other sources, such as an Apex application or a SOAP API call, Salesforce validates only the foreign keys. Before executing a trigger, Salesforce verifies that any custom foreign keys do not refer to the object itself.
   Salesforce runs user-defined validation rules if multiline items were created, such as quote line items and opportunity line items.
3. Executes all before triggers.
4. Runs most system validation steps again, such as verifying that all required fields have a non-null value, and runs any user-defined validation rules. The only system validation that Salesforce doesn't run a second time (when the request comes from a standard UI edit page) is the enforcement of layout-specific rules.
5. Executes duplicate rules. If the duplicate rule identifies the record as a duplicate and uses the block action, the record is not saved and no further steps, such as after triggers and workflow rules, are taken.
6. Saves the record to the database, but doesn't commit yet.
7. Executes all after triggers.
8. Executes assignment rules.
9. Executes auto-response rules.
10. Executes workflow rules.
11. If there are workflow field updates, updates the record again.
12. If the record was updated with workflow field updates, fires before update triggers and afterupdate triggers one more time (and only one more time), in addition to standard validations. Custom validation rules, duplicate rules, and escalation rules are not run again.

    The refiring of triggers isn't limited to updates, but applies to all operation types. A workflow field update that fires on record insert will rerun any before and after insert

triggers again—as insert triggers.

13. Executes processes and flows launched via processes and flow trigger workflow actions. When a process or flow executes a DML operation, the affected record goes through the save procedure.
14. Executes escalation rules.
15. Executes entitlement rules.
16. If the record contains a roll-up summary field or is part of a cross-object workflow, performs calculations and updates the roll-up summary field in the parent record. Parent record goes through save procedure.
17. If the parent record is updated, and a grandparent record contains a roll-up summary field or is part of a cross-object workflow, performs calculations and updates the roll-up summary field in the grandparent record. Grandparent record goes through save procedure.
18. Executes Criteria Based Sharing evaluation.
19. Commits all DML operations to the database.
20. Executes post-commit logic, such as sending email.

During a recursive save, Salesforce skips steps 8 (assignment rules) through 17 (roll-up summary field in the grandparent record).

Additional Considerations

Note the following when working with triggers.

- The order of execution isn't guaranteed when having multiple triggers for the same object due to the same event. For example, if you have two before insert triggers for Case, and a new Case record is inserted that fires the two triggers, the order in which these triggers fire isn't guaranteed.
- When a DML call is made with partial success allowed, more than one attempt can be made to save the successful records if the initial attempt results in errors for some records. For example, an error can occur for a record when a user-validation rule fails. Triggers are fired during the first attempt and are fired again during subsequent attempts. Because these trigger invocations are part of the same transaction, static class variables that are accessed by the trigger aren't reset. DML calls allow partial success when you set the allOrNone parameter of a Database DML method to false or when you call the SOAP API with default settings. For more details, see Bulk DML Exception Handling.
- If your org uses Contacts to Multiple Accounts, anytime you insert a non-private contact, an AccountContactRelation is created and its validation rules, database insertion, and triggers are executed immediately after the contact is saved to the database (step 6). When you change a contact's primary account, an AccountContactRelation may be

created or edited, and the AccountContactRelation validation rules, database changes, and triggers are executed immediately after the contact is saved to the database (step 6).

- If you are using before triggers to set Stage and Forecast Category for an opportunity record, the behavior is as follows:
- If you set Stage and Forecast Category, the opportunity record contains those exact values.
- If you set Stage but not Forecast Category, the Forecast Category value on the opportunity record defaults to the one associated with trigger Stage.
- If you reset Stage to a value specified in an API call or incoming from the user interface, the Forecast Category value should also come from the API call or user interface. If no value for Forecast Category is specified and the incoming Stage is different than the trigger Stage, the Forecast Category defaults to the one associated with trigger Stage. If the trigger Stage and incoming Stage are the same, the Forecast Category is not defaulted.
- If you are cloning an opportunity with products, the following events occur in order:

1. The parent opportunity is saved according to the list of events shown above.
2. The opportunity products are saved according to the list of events shown above.

   If errors occur on an opportunity product, you must return to the opportunity and fix the errors before cloning.
   If any opportunity products contain unique custom fields, you must null them out before cloning the opportunity.

- Trigger.old contains a version of the objects before the specific update that fired the trigger. However, there is an exception. When a record is updated and subsequently triggers a workflow rule field update, Trigger.old in the last update trigger doesn't contain the version of the object immediately before the workflow update, but the object before the initial update was made. For example, suppose that an existing record has a number field with an initial value of 1. A user updates this field to 10, and a workflow rule field update fires and increments it to 11. In the update trigger that fires after the workflow field update, the field value of the object obtained from Trigger.old is the original value of 1, rather than 10, as would typically be the case.
- The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use the Flows action in Process Builder instead.

From <https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_order_of_execution.htm>