Validation Rules

- This is a declarative feature to restrict user entering invalid data.
- Used to set custom error message when the input data is not in defined format
- Used to make a field required conditionally.
- This will fire only when new record is inserted / updated.
- When the expression returns true, then error message will be thrown.

Two Types

1. Standard Validation

2. Custom Validation

When it fires?

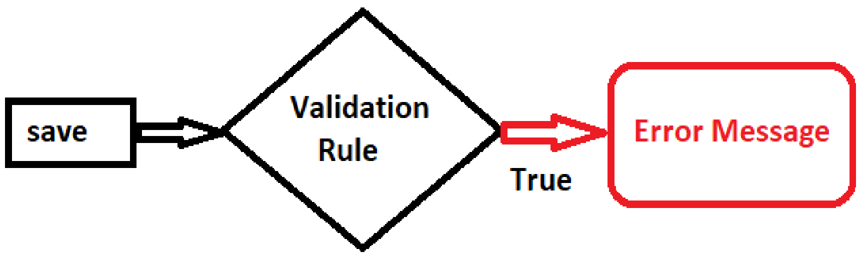
When the record is saved (due to insert or modification)

Why should I need validation rules?

i. To make a field conditionally required.
ii. To make a field required
iii. To accept the input in certain defined format
iv. To set a field as required such as CheckBox which is not possible in edit page of field or page layout.

Validation rule is like a formula consists of Expression.
This will always return Boolean (True or False).

It works as below. When the record is saved, then validation rules will be invoked similar to standard validations. When the rules resolved to true, then error message will be thrown in the interface.



This custom error message will be displayed next to the field optionally and at the top of the page.

Error message will be thrown in interface if user saved the record using standard layout. If the record is inserted or modified through API, then the error message will be thrown in system log or error file.

Use Case 1: The course begins should NOT be greater than Course End.
How to create validation rule for the above use case?
Navigation:
Setup-> Build-> Create-> Objects, then choose your object
Step 1: Go to Validation Rule section in Object Definition page
Step 2: Click New and enter the rule name
Enter the expression to execute
Enter the error message.

Fields in formula editor

i. Can be included from current object and related object.
ii. From global variables such as currently logged in user, profile, roles etc.,

For this use case, the rule will be as follows
IF(Course_Start_Date__c > Course_End_Date__c , true, false)

Use Case 2:
When the status is in progress, then course fee is mandatory
Use Case 3:
When Virtual and Class Room types are enabled, then Contact NO must be given.
Use Case 4:
When the status is approved, then should not allow user to modify the course fee.

<u>SOLUTION FOR USE CASES 2:</u>
Approach to build the validation rules.
Considering "When error message thrown"
Step 1: Break the requirement into running sentence
Step 2: Replace with Actual Field and Operators if applicable
Step 3: Combine into one formula

When to throw the error message.
Step 1: Running sentence
i. the status is in progress
and
ii. When course fee is space

Step 2:
i. Status__c = 'IN Progress'
and
ii. Course_Fee__c = ' '

Step 3:
i. TEXT(Status__c) = 'In Progress' | ISPICKVAL(Field, Value) => return when field = value
ISPICKVAL(Status__c, 'In Progress')
and
ii. ISBLANK(Field) => returns true when the field is blank
ISBLANK(Course_Fee__c)

Step 4: Final Formula
AND(Cond1, Cond2...) => True when all the conditions are met.
Final Formula: AND(ISPICKVAL(Status__c, 'In Progress'), ISBLANK(Course_Fee__c))

Solution for Use Case 3A:
When Virtual and Class Room types are enabled, then Contact NO must be given.

Step 1: Running sentence - When to throw the error message.
i. Virtual is true and Class Room is true
and
iii. Contact No is blank

Step 2:
i. Virtual__c = True and Class_Room__c = True
               and
iii. ISBLANK(Contact_No__c)

Step 3: since both the conditions need to be met, user AND function. **Final Formula.**
AND(Virtual__c, Class_Room__c, ISBLANK(Contact_No__c))

Solution for Use Case 3B:
When Virtual or Class Room type is enabled, then Contact NO must be given.

Step 1:
i. Virtual__c = True or Class_Room__c = True and
iii. ISBLANK(Contact_No__c)
Step 2:
OR(cond1, cond2..) => true when any one of the given argument is true.
OR(Virtual__c, Class_Room__c)
             and
ISBLANK(Contact_No__c)
Step 3: Final Formula
AND(OR(Virtual__c, Class_Room__c) , ISBLANK(Contact_No__c))

Use Case 4:
When the status is approved, then should not allow user to modify the course fee.
Step 1:
i. the status is approved
          and
ii. when course fee is modified. There is a inbuilt function to check if the field is modified or not as below.

ISCHANGED(field) => return true when the argument is modified.
Step 2:
i. ISPICKVAL(Status__c, 'Approved')
          and
ii. ISCHANGED(Course_Fee__c)

Step 3: Final Formula
AND(ISPICKVAL(Status__c, 'Approved'), ISCHANGED(Course_Fee__c))

Use Case 6:
Should not allow user to add new employee for the departments which are not approved or obsolete.
Use Case 7: (Exercise)
Last working day value must be given if the employee is in notice period. (Custom field 'Last working day ' must be added of date type)
Use Case 8:
Do not allow to change the status from resigned to any other status.

<u>**Solution for Use Case 6**</u>
Should not allow user to add new employee for the departments which are not approved or obselete.
Step 1: Running sentence - When to throw the error message.

i. When new record is added
    and
ii. Department status = approved or obselete

Step 2:
i. ISNEW() => This is an inbuilt function used to check if when the record is new
    and
ii. ISPICKVAL(Department__r.Status__C, 'Not Approved') Or
   ISPICKVAL(Department__r.Status__C,'Obselete')

Step 3:
AND(ISNEW(), OR(ISPICKVAL(Department__r.Status__C, 'Not Approved'),
ISPICKVAL(Department__r.Status__C,'Obselete')))

Note: Formula which is referring the field from parent object is called CROSS Object formula
Use Case 8:
Do not allow to change the status from resigned to any other status.

ISPICKVAL(Field,value) => return if field = value

Example:
ISPICKVAL(Status__C, 'Resigned')

PRIORVALUE(field) => this will return the current value.
Step 1:
ISPICKVAL(PRIORVALUE(Status__c) , 'Resigned')
    and
ISCHANGED(field) => return true when the field in this argument is modified.

Step 2: Final Formula
AND(ISPICKVAL(PRIORVALUE(Status__c) , 'Resigned'), ISCHANGED(Status__C))
condition 1   **condition 2**

Ranjith Krishnan
sfdcmeet@gmail.com