BUILDING APPLICATIONS WITH FORCE.COM – PART 2 (PRESERVING DATA QUALITY)

Exercise Guide

Table of Contents

11-1: Create Validation Rules 1
11-2: Build Validation Rules to Enforce Conditionally Required Fields 2
11-3: Build Validation Rules to Enforce Data Format 4
11-4: Build Validation Rules to Enforce Consistency 6
11-5: Create Validation Rules to Prevent Data Loss (Optional) 9

11-1: Create Validation Rules

Scenario:

Universal Containrs (UC) employees should not be able to save a position record unless the Hiring Manager field is filled out.

Goal:

Create validation rules to enforce business requirements.

Task:

Create a new validation rule that requires that all positions must have a Hiring Manager listed.

Time:

10 minutes

Instructions:

1. Create a new validation rule that requires that all positions must have a Hiring Manager listed.
   A. Click **Setup | Create | Objects | Position**.
   B. In the Validation Rules related list, click **New**.
      i. **Rule Name**: `Every Position Must Have a Hiring Mgr`
      ii. **Active**: (selected)
      iii. **Description**: `Every position record must have a hiring manager.`
      iv. **Error Condition Formula**: `ISBLANK( Hiring_Manager__c )`
      v. Click **Check Syntax** to verify your formula.
      vi. **Error Message**: `Every Position must have a Hiring Manager.`
      vii. **Error Location**: `Field: Hiring Manager`
   C. Click **Save**.
   D. Create a new Position record to test these validation rules.

11-2: Build Validation Rules to Enforce Conditionally Required Fields

Scenario:

Universal Containers would like to enforce its policies around Temporary positions. The Duration field on a Temp position should not be blank. It should contain a value between 1 and 365.

Goal:

Build a validation rule that prevents users from saving Temp positions with a blank Duration.

Tasks:

1. Build a new validation rule that ensures that these policies are followed.
2. Set the Debug Log to track actions that you take.
3. Create a new position to test that the validation rule works.
4. Check the Debug Logs.

Time:

10 minutes

Instructions:

1. Build a new validation rule that ensures that these policies are followed.
   A. Click **Setup | Create | Objects | Position**.
   B. Scroll down to the Validation Rules related list and click **New**.
      i. **Rule Name**: `Temp Position Validation`
      ii. **Active**: (selected)
      iii. **Description**: `Temporary positions require a value for Duration between 1 and 365 days.`
      iv. **Error Condition Formula**: `ISPICKVAL( Type__c , "Temp") && (BLANKVALUE(Duration__c,0)<1 || Duration__c >365)`
      v. Click **Check Syntax** to verify your formula.
      vi. **Error Message**: `Temporary positions require a value for Duration between 1 and 365 days.`
      vii. **Error Location**: `Field: Duration`
   C. Click **Save**.
2. Set the Debug Log to track actions that you take.
   A. Click **Setup | Monitor | Logs | Debug Logs**.
   B. Click **New** in User Trace Flags.
   C. Click the lookup icon to search for and then select your name.

D. Set up a new Debug Level.
   i. Click the lookup icon to select Debug Level.
   ii. Click **New**.
   iii. Type `Test` in the **Name** field.
   iv. Click **Save**.

E. Click **Save**.

3. Create a new position to test that the validation rule works.
   A. Click the Positions tab.
   B. Click **New**.
   C. Select `Non-Technical Position` from the **Record Type of new record** picklist, and click **Continue**.
      i. **Title**: `Assistant to the Director of Support`
      ii. **Type**: `Temp`
      iii. **Department**: `Support`
      iv. **Location**: `San Francisco`
      v. **Pay Grade**: `S-100`
      vi. **Hiring Manager**: `Frank Linstrom`
      vii. **Priority**: `High`
      viii. **Status**: `New`
      ix. **Date Opened**: *`(today's date)`*
      x. **Job Description**: `The Assistant to the Director of Support is a diverse and fast-paced role supporting the director of our 250 person support organization.`
   D. Click **Save**.
   E. After receiving the error `Temporary positions require a value for Duration between 1 and 365 days`, enter the Duration as `364`, then click **Save** again.

4. Check the Debug Logs.
   A. Click **Setup | Monitor | Logs | Debug Logs**.
   B. In the section called Debug Logs, click the **View** link next to the earliest listing (at the bottom of the list) for your name to view the logs.

      Note the result: VALIDATION_FAIL

   C. Click **Back to List: Debug Logs** (in the upper left).
   D. Click the **View** link next to the latest listing for your name to view the logs.

      Note the result: VALIDATION_PASS

11-3: Build Validation Rules to Enforce Data Format

Scenario:

Universal Containers would like to make sure that when candidates are entered, the zip code is entered in the correct format.

Goal:

Build a validation rule that enforces proper data format.

Tasks:

1. Create a validation rule on candidates that requires that zip codes be entered in a valid 5-digit or 9-digit format.
2. Test the validation rule on an existing candidate.

Time:

10 minutes

Instructions:

1. Create a validation rule on candidates that requires that zip codes be entered in a valid 5-digit or 9-digit format.
   A. Click **Setup | Create | Objects | Candidate**.
   B. Scroll down to the Validation Rules related list and click **New**.
      i. **Rule Name**: `Zip code must be Valid US Postal Code`
      ii. **Active**: (selected)
      iii. **Description**: `Validates that the candidate Zip/Postal Code is in 99999 or 99999-9999 format if Country is USA or US.`
      iv. **Error Condition Formula**: `REGEX(UPPER(Country__c), "USA?") && NOT(REGEX(Zip_Postal_Code__c, "\\d{5}(-\\d{4})?"))`
      v. Click **Check Syntax** to verify your formula.
      vi. **Error Message**: `Zip code must be in 99999 or 99999-9999 format.`
      vii. **Error Location**: `Field: Zip/Postal Code`
   C. Click **Save**.
2. Test the validation rule on an existing candidate.
   A. Click the Candidates tab.
   B. Select `All` from the **View** picklist and click **Go!**
   C. Select any `Candidate` and click **Edit**.
      i. **Country**: `USA`
      ii. **Zip/Postal Code**: `9410`

D. Click **Save**.
E. After receiving the error `Zip code must be in 99999 or 99999-9999 format`, update the zip code to `94105` and click **Save**.

11-4: Build Validation Rules to Enforce Consistency

Scenario:

Universal Containers would like to ensure that when a zip code is entered, it matches the state that's entered. For example, a candidate with a California zip code should not have a state of New York. This rule should be ignored when data is loaded in batch.

Goal:

Build a validation rule that enforces data consistency.

Tasks:

1. Create an object on which to store zip code data.
2. Create additional fields on the Zip Code object.
3. Create new zip code records.
4. Create a validation rule that checks the zip code entered against a table to validate that the zip code and state match.
5. Create a new Candidate to test your Zip Code object and validation rule.

Time:

15 minutes

Instructions:

1. Create an object on which to store zip code data.
    A. Click **Setup | Create | Objects**.
    B. Click **New Custom Object**.
        i. **Label**: `Zip Code`
        ii. **Plural Label**: `Zip Codes`
        iii. **Object Name**: `Zip_Code` (This field auto-populates.)
        iv. **Context-Sensitive Help Setting**: `Open the standard Salesforce.com Help & Training window.`
        v. **Record Name**: `Zip Code`
        vi. **Data Type**: `Text`
        vii. **Allow Reports**: (selected)
        viii. **Allow Activities**: (cleared)
        ix. **Track Field History**: (cleared)
        x. **Allow Search**: (selected)
        xi. **Deployment Status**: `Deployed`
        xii. **Launch New Custom Tab Wizard after saving this custom object**: (selected)
    C. Click **Save**.
    D. Use the lookup icon to select the Map tab style, and click **Next**.

E. Make the tab `Default On` for the Custom-HR, Custom-Executive, Custom-Recruiter and System Administrator profiles and click **Next**.
F. Include the tab in the Recruiting application only, and click **Save**.

2. Create additional fields on the Zip Code object.

A. Under Custom Fields & Relationships, click **New**.
B. Select the `Text` radio button and click **Next**.
C. Enter the details for the first new custom field.

i. **Field Label**: `State`
ii. **Length**: `2`
iii. **Field Name**: `State` (This field auto-populates.)

D. Click **Next**.
E. Set the field visible for the Custom-HR, Custom-Executive, Custom-Recruiter and System Administrator profiles and click **Next**.
F. Click **Save & New** to add the field to the zip code layout.
G. Select the `Text` radio button and click **Next**.
H. Enter the details for the second new custom field.

i. **Field Label**: `City`
ii. **Length**: `80`
iii. **Field Name**: `City` (This field auto-populates.)

I. Click **Next**.
J. Set the field visible for the Custom-HR, Custom-Executive, Custom-Recruiter, and System Administrator profiles, and click **Next**.
K. Click **Save** to add the field to the zip code layout.
L. Modify the Zip Code page layout so that the fields are displayed in a logical order.

i. Under the Page Layout related list, click the **Edit** link next to Zip Code Layout.
ii. Arrange the fields in the left hand column of the page layout so that **City** is on top, then **State,** then Zip Code.
iii. Click **Save**.

3. Create new zip code records.

A. Click the Zip Codes tab.
B. Click **New**.
C. Enter the **City, State,** and **Zip Code** for any location.
D. Click **Save**.
E. Repeat to create 3 to 5 zip code records (to look up Zip Codes for any city, go to `http://zip4.usps.com/zip4/citytown.jsp`).

4. Create a validation rule that checks the zip code entered against a table to validate that the zip code and state match.

A. Click **Setup | Create | Objects | Candidate**.
B. Scroll down to the Validation Rules related list and click **New**.

  i. **Rule Name**: `Zip Code Consistent with State`
  ii. **Active**: (selected)
  iii. **Description**: `Validates candidate Zip/Postal Code by looking up the first five characters of the value in a custom object called Zip_Code__c. Error if the zip code is not found or the candidate State does not match the corresponding State in the object.`
  iv. **Error Condition Formula**:

```
AND(VLOOKUP($ObjectType.Zip_Code__c.Fields.State__c,
$ObjectType.Zip_Code__c.Fields.Name, LEFT( Zip_Postal_Code__c
,5) ) <> State_Province__c, NOT(Batch_Load_Item__c) )
```

  v. Click **Check Syntax** to verify your formula.
  vi. **Error Message**: `Candidate Zip Code does not exist in specified State.`
  vii. **Error Location**: `Field: Zip/Postal Code`

C. Click **Save**.

5. Create a new Candidate to test your Zip Code object and validation rule.

A. Click the Candidates tab.
B. Click **New**.

  i. Populate a new Candidate with a **Zip Code** that matches a Zip Code record that you have already created.
  ii. Enter a **State** that does not match the Zip Code.

C. Click **Save**.
D. Note the validation error that you receive.

11-5: Create Validation Rules to Prevent Data Loss (Optional)

Scenario:

Universal Containers would like make sure that once a job application is approved, users will not be able to add or remove reviews.

Goal:

Build a validation rule that prevents users from adding or deleting reviews once a job application has been approved.

Tasks:

1. Create a validation rule that references the Total Reviews roll-up summary field to ensure that reviews are not added or deleted.
2. Test the validation rule.

Time:

10 minutes

Instructions:

1. Create a validation rule that references the Total Reviews roll-up summary field to ensure that reviews are not added or deleted.
   A. Click **Setup | Create | Objects | Job Application**.
   B. Scroll down to the Validation Rules related list and click **New**.
      i. **Rule Name**: `No New Deleted Reviews for Approved Apps`
      ii. **Active**: (selected)
      iii. **Description**: `Once a Job Application is approved, there can be no new Reviews. Likewise, no Reviews can be deleted.`
      iv. **Error Condition Formula**:
      ```
      AND( ISPICKVAL( Status__c , "Approved") ,  ISCHANGED(
      Total_Reviews__c ))
      ```
      v. Click **Check Syntax** to verify your formula.
      vi. **Error Message**: `Once a Job Application is approved, there can be no change to the number of Reviews.`
      vii. **Error Location**: `Top of Page`
   C. Click **Save**.
2. Test the validation rule.
   A. Click the Job Applications tab.
   B. Click **APP-0000**.

C. Click **Edit**.
   i. Change the **Status** to `Approved`.
   ii. Click **Save**.
D. Scroll down and click the **Del** link next to the review listed in the Reviews related list.
E. When you receive the popup that says, “Are you sure?” click **OK**.