Data Validator - Setting up Apex Rules

Introduction

This document aims to explain how to setup apex validation rules to be used with the Data Validator.

dv_ApexValidationInterface

The ***sfdmgr.dv_ApexValidatorInterface*** is a global class included within the Salesforce Data Manager package. This will be the extension gateway for the package to invoke your class and methods for validation rules. Your class that extends this interface should be stored on your Data Validation Instance record in the ***sfdmgr__Apex_Validation_Class__c*** field.

```
/**
 * Created by dgajwani on 3/19/20.
 */

global with sharing class accountValidator extends sfdmgr.dv_ApexValidationInterface{

    global override virtual Map<String, Object> processValidation(Map<String, Object>
        sfdmgr.Logger.log('M:E', 'accountValidator:processValidation');
        // Return a Map from this class
        sfdmgr.Logger.log('M:X', 'accountValidator:processValidation');
    }
}
```

processValidation(Map<String,Object> inputData)

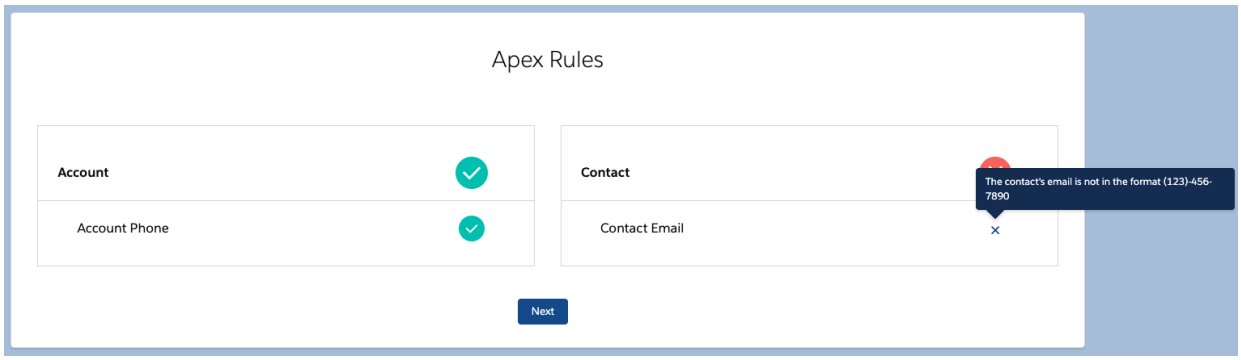
The processValidation method is offered as a part of the dv_ApexValidationInterface. This is the method that you should be overriding to return your results. The method takes a Map<String,Object> as a parameter that contains the following:

1. The current Data Validator Instance SFID.
2. Map of instance variables and their values.

Example Input

```
{
    "currentInstanceName" : "01I4P000000h6wr",
    "instanceVariablesMap" {
        "accountSFID" : "0014P000026w58E"
    }
}
```

Understanding how the UI is rendered



To render the above UI, the following output map is expected from your ***processValidation()*** method:

```
{
  "cardsList": [          // Use this list to control the sequence of your cards.
    "Account",
    "Contact"
  ],
  "cardsDataMap": {
    "Account": {
      "rulesList": [     // Use this list to control the sequence of your rules.
        "Account Phone"
      ],
      "rulesMap": {
        "Account Phone": {
          "Success": true
        }
      },
      "OverallResult": true
    },
    "Contact": {
      "rulesList": [
        "Contact Email"
      ],
      "rulesMap": {
        "Contact Email": {
          "ErrorMessage": "The contact's email is not in the format (123)-456-7890",
          "Success": false
        }
      },
      "OverallResult": false
    }
  },
  "OverallResult": false
}
```

To avoid making spelling mistakes (we all do it at some point!), the package provides the following global constants to make your life easier:

Constant	Text
sfdmgr.dv_Constants.*OVERALL_RESULT*	OverallResult
sfdmgr.dv_Constants.*RESULT_ERROR_MESSAGE*	ErrorMessage

sfdmgr.dv_Constants.*RULES_MAP*	rulesMap
sfdmgr.dv_Constants.*RULES_LIST*	rulesList
sfdmgr.dv_Constants.*CARDS_DATA_MAP*	cardsDataMap
sfdmgr.dv_Constants.*CARDS_LIST*	cardsList

Logging

The package provides a robust logger so that you do not have to setup debug traces or contact Support! Use the following methods in your class to print logs directly in the console:

```
global class Logger {

    global static void log(final String message)

    global static void log(final String subject, final String message)

    global static void log(final System.LoggingLevel loggingLevel, final String subjec

}
```

Use LoggingLevel to color code your log!

System.LoggingLevel.INFO

```
[1585440962766] | [INFO] | M:X | dv_Util:object2FieldsMap
```

System.LoggingLevel.WARN

```
[1585440962805] | [WARN] | Contact | Skipping object since query has missing merge variables.
```

System.LoggingLevel.ERROR

```
[1585441100969] | [ERROR] | Runtime Exception: | Error querying object.
[1585441100969] | [ERROR] | Stack Message | unexpected token: 'WHERE'
```