Module 6 Lab C

Comparing Control and Form Notifications

Objective

Use field and form level validation, display error notifications at the control and form level, compare the differences, and turn off auto-save.

Scenario

The new\_bankaccount entity must validate the entry of location and sortcode, UK must have 8 digit codes starting with 044, USA must have 8 digit account codes starting with 01. Use field level Notification to display against the sortcode, demonstrate that rule infringement prevents the record being saved. Use the same logic but with Form Notification to show that now the record can be saved whilst the record breaks the validation rule, and the message disappears on form refresh. Attach an event handler to the form\_onsave, call the validation, and prevent the default save behaviour if the data is not valid. The validation must not use modal alerts.

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| 1 | Open the Lab6cStart.js file located in Module 6 Lab C starter folder in Visual Studio.  This starter script registers a function, ValidateForm (not written yet), to execute when the location or sortcode fields are modified. |
| 2 | Create a function ValidateForm (), that creates variables location and code, and assign these with the values of the new\_banklocation, and new\_sortcode fields. Create a variable to point to the sortcode control. Declare a variable named result that is initialised to false. |
| 3 | Following that within the function check to see if the location and code are not null.  If this is the case then check the SortCode is valid by passing the sortcode and the location to a Boolean returning function named ValidSortCode.  if (!QALab.ValidSortCode(sortCode, location)) {}  If not valid then display a Field Notification against the new\_sortcode.  sortCodeCtrl.setNotification("The account code does not match the selected location."); |
| 4 | Create a ValidSortCode function with parameters sortcode and location. In the method define a variable for a regular expression pattern (named pattern) and assign it \\d{8}. Perform a switch on the location,toLowerCase(), and if “uk” assign the pattern to “^044[1-9]{5}$”, if location is usa, assign the pattern to “^01[1-9]{6}$”. After the switch return the result of a new regular expression that test the sortcode against the pattern  QALab.ValidSortCode = function (sortCode, location) {  var pattern = "\\d{8}";  console.log("sortCode is " + sortCode);  console.log("location is " + Location);  switch (location.toLowerCase()) {  case "uk":  pattern = "^044[1-9]{5}$";  break;  case "usa":  pattern = "^01[1-9]{6}$";  break;  default:  break;  }  var RegExpPattern = new RegExp(pattern);  return RegExpPattern.test(sortCode);  }; |
| 5 | The completed function should look like this |
| 6 | Add the Lab6cStart.js as a web resource to CRM, and customise the bank account form, adding Lab6c.js as a JavaScript library and associating the QALab.OnFormLoad event with the form load event. Test the validation logic, and notice that the records with bad values cannot be saved. |
| 7 | Modify the field notifications to form notifications. In the ValidateForm method change the script in the conditional block to use Form Notifications.  if (!QALab.ValidSortCode(sortCode, location)) {  console.warn("sortCode is not correct");  Xrm.Page.ui.setFormNotification("The account code does not match the selected location.", "ERROR", "InvalidSortCode");    }  else {  Xrm.Page.ui.clearFormNotification("InvalidSortCode");  } |
| 8 | Upload the mofified javascript file as a webresource and associate the script with the BankAccount entity's main form in a similar way to the previous instructions.  Note that records can now be saved with bad values. |
| 9 | We will now fix that issue. Modify the ValidateForm method to return true or false depending on whether the sortcode is valid.  Add an event handler to attach to the save event called OnFormSave\_BankAccount with a parameter eventContext.  Create a variable eventArgs and assign it from eventContext.getEventArgs().  Check to see that the eventArgs.getSaveMode() method returns 70 , 1 or 2 (auto, manual save or Save and close)  if so call the ValidateForm method and prevent the save if not valid, by eventArgs.preventDefault(). |
| 10 | The code should look like this  QALab.ValidateForm = function () {  var sortCode = Xrm.Page.getAttribute("new\_sortcode").getValue(); // get a refernce to the field that stores the data  var sortCodeCtrl = Xrm.Page.getControl("new\_sortcode"); // get a reference to the control  var location = Xrm.Page.getAttribute("new\_banklocation").getValue();  var result = false;  if (sortCode != null && location != null) {  console.warn("not null");  if (!QALab.ValidSortCode(sortCode, location)) {  console.warn("sortCode is not correct");  //sortCodeCtrl.setNotification("The account code does not match the selected location.");  Xrm.Page.ui.setFormNotification("The account code does not match the selected location.", "ERROR", "InvalidSortCode")  result = true;  }  else {  // sortCodeCtrl.clearNotification();  Xrm.Page.ui.clearFormNotification("InvalidSortCode");  result = false;  }  }  return result;  }; |
|  | QALab.OnFormSave\_BankAccount = function (econtext) {  var eventArgs = econtext.getEventArgs();  if (eventArgs.getSaveMode() == 70 //autosave  || eventArgs.getSaveMode() == 1 // manual save  || eventArgs.getSaveMode() == 2) //save and close  {  if (QALab.ValidateForm()) {  //Prevent auto/manual save  eventArgs.preventDefault();  }  }  } |
| 11 | Now upload the modified JavaScript library as a webresource and associate it in to the BankAccount entity registering 12 the OnLoad and OnSave events against the relevant functions in the script file.     1. Note that the function that handles the Save event needs to be passed execution context. See following diagram.      1. Test that the latest version of the script works as expected. |