Module 7 Lab C

Using WebResources

Objective: Add HTML pages and scripts as webresources to customise the Account entities information form

Scenario

We would like users to be able to select existing contacts to be associated with accounts. Users should be able to select from a list of contacts based on their location.

Step1: Adding the Webresources

|  |  |
| --- | --- |
| 1 | Open the Visual Studio Solution named EmptyWebApp located in the Starter folder for this Lab. There are two html pages located in the HTML folder. One of these, named "AddExistingContac.html" will render a button which will be displayed in the Account information form as follows |
| 2 | When the "select existing contact" button is clicked the second page "ContactSelector.html" will render as follows. This page allows contacts to be listed based on location. The selected contact can then be added to the contacts associated with the account. |
| 3 | Open both of these files in the editor and you will see that they reference three scripts. |
| 4 | The first reference, ClientGlobalContext.js.aspx, will allow us to access a call context object. The context object is required by the SDK.REST.js script which has been copied from the Dynamics 365 SDK |
| 5 | The last script LAB7B.js is one that is yet to be completed but has some existing functions including one that will allow us to retrieve querystring parameter information. |
| 6 | Notice that the AddExistingContact.html page makes a call to a function named ShowContactSelector  <input onclick="ShowContactSelector();" type="button" value="Select Exisiting Contact"> |
| 7 | Add a parameter-less function named ShowContactSelector to the LAB7B.js file. Call the getMyQueryString function to retrieve the value associated with the id parameter and store a decoded version in a local variable named accountId |
| 8 | Use the XrmUtility.openWebResource function to display the second web resource which will named new\_/html/ContactSelector.html, passing the accountId as the "Data" parameter and sizing the window at 500 x 300 |
| 9 | Let's test that this works by uploading the LAB7B.js, SDK.REST.js scripts and the two html pages as webresources to the Module 7 solution in Dynamics 365 as follows. |
| 10 | Remember to save and publish all the customisations. |
| 11 | We will now customise the Account entity so that someone using the main form will see our customisations.  First Add the Account Entity and all its associated assets to the Module 7 solution. |
| 12 | Expand the Account entity and select to edit the form named "Account". |
| 13 | Place your cursor below the contacts sub grid as shown below    And then |
| 14 | Add a webresource using the Insert tab at the top of the form  Make sure you select the pass record object type code and unique identifier as parameters option |
| 15 | Save and publish your Customisation. |
| 16 | Navigate to the Sales area and open up the information form for an existing Active Account. Scroll down to the section you created above the existing contacts section and test that the Add existing contact button appears and that it opens a new window when clicked. Only the Country list should be populated now. |

Step 2 : Using the WeApi endpoint to retrieve and update data

|  |  |
| --- | --- |
| 1 | In the LAB7B.js file add a global variable named AccountID and a parameter-less function named GetParams. Note that the SelectContact.html page makes a call to the GetParams() function |
| 2 | The Parameter in this lab will be passed through one named "Data", but in a later lab it will be passed via a parameter named "id".  Use the getMyQueryString function to assign a decoded entity identifier to the AccountID variable whether the parameter is named Data or id |
| 3 | The second function, GetContacts, called when the ContactSelector page is loaded, will need to retrieve a list of contacts based on the country selected in the countries list. |
| 4 | Below the following comment  //add code to retrieve contacts here  add a variable named querystring. This will be assigned a ODATA query that will restrict Contact to ones that are based on country |
| 5 | The starter project has a file in the scripts folder named JavaScriptRESTDataOperationsSample.js (taken from the SDK). Open it and Explore some of the functions that call SDK.REST.js functions to create, read, update and delete dynamics 365 data.  e.g. The function named getFirstContactToBePrimaryContact calls the function SDK.REST.retrieveMultipleRecords use it as a template to implement the rest of the GetContacts method. |
| 6 | The querystring variable in our application will execute an ODATA request that should return the ContactId and FullName of the contact, filtered on the Country the contact is in and should restrict the results to at most 10 |
| 7 | Use the $select, $filter and $top ODATA query operators to generate the correct string and assign it to the querystring variable. |
| 8 | Use this string in the call to RetrieveMultiple and on success call the DisplayContacts function passing the results of the query. On failure invoke the errorhandler. |
| 9 | Save your work and then upload the modified script to the webresource new\_/scripts/LAB7B.js |
| 10 | Save and publish the customisation and then open up an active account and test that the add existing contact button opens a window that now allows you to select contacts based on country. |
| 11 | Now we need to associate the selected contact with the account.  The SDK.REST.js library has a function named associateRecords that accepts parameters parent record identifier, parent type name and relationship name, as well as the identifier and type name of the child entity.  SDK.REST.associateRecords(AccountID, "Account", "contact\_customer\_accounts", contactID, "Contact", function () { alert('Success') }, errorHandler); |
| 12 | Add a parameter-less function to the LAB7B.js file named AssociateToAccount that retrieves the value of the item selected in the contacts list and uses it to associate the contact to the account. |
| 13 | This function will get called when the "Associate to Account" button is clicked |
| 14 | Save your changes and upload the new version to your Module 7 solution save and publish the changes and test that it works by attempting to add an existing contact to an existing account.    The contacts sub grid does not automatically refresh. Click on the "Full Name" column header to see the contact that has been added. |