Lightning Web Component

Dynamic custom lookup in LWC

The Context

Term 'Lookup' in Salesforce is used to look or search for existing record of specific type in the system. Custom lookup in SLDS provides same look and feel of any lookup field on record detail page. In this blog we are going to see how we can create single dynamic lookup component **which will look up to more than one SObject**, you just have to pass array of Object(s), list of fields to be displayed when result is found and filter criteria if any.

Approach

1. SLDS provides look and feel for Lookup functionality, Please see in references section.

2. Our component is dynamic and can be placed in any component hence it acts as a Child component which accepts inputs from Parent component.

3. **Component overview**

a) Below format is required as an input to our dynamic component, this input is in format of 'array of objects' defined in parent's JS.

b) **As you can see we have passed multiple object data, our component is built to provide you lookup result for more than one SObject.**

```
        //You can also use Custom Metadata Type to store below
configuration
        ObjectConfig = [ //Array of objects
            {
                    'label':  'Contact',
                    'APIName': 'Contact',
                    'fields':'Name,FirstName,LastName,Email,Phone',
                    'displayFields':'Name,Phone,Email',
                    'iconName': 'standard:contact',
                    'FilterCondition' : 'AccountId != NULL'
            },
            {
                    'label':  'Account',
                    'APIName': 'Account',
                    'fields':'Name',

    'displayFields':'Name,AnnualRevenue,AccountNumber',
                    'iconName': 'standard:account',
                    'FilterCondition' : 'AccountNumber != NULL'
            }
        ];
```

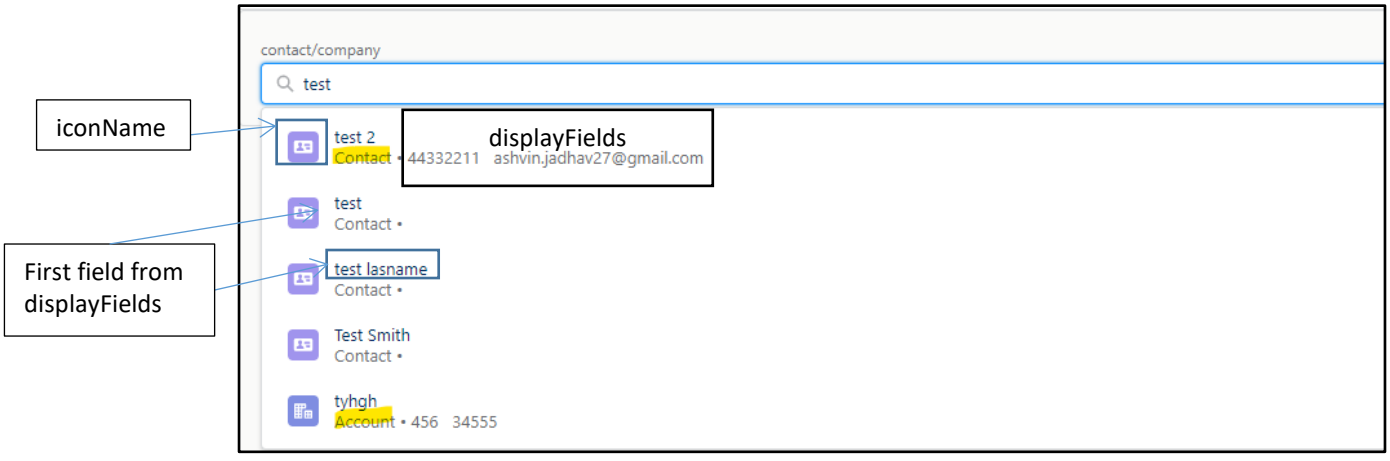
c) Our dynamic component's controller will parse this array and return the JSON in below format. JSON will include all records returned by SOSL.

codebeautify.json

d) JSON is created in Apex using **JSON.createGenerator(true)**

```
74          //Traverse through each returned record and generate JSON format
75          JSONGenerator jsonGen = JSON.createGenerator(true);
76          jsonGen.writeStartObject();
77          for(List<sObject> records : searchRecords){
78              jsonGen.writeFieldName(string.valueof(records.getSObjectType()));
79              jsonGen.writeStartArray();
80              String[] displayFieldLst = mapofobjDisplayFields.get(string.valueof(records.getSObjectType())).split(',');
81              for(sobject record : records){
82                  jsonGen.writeStartObject();
83                  jsonGen.writeFieldName('displayFields');
84                  jsonGen.writeObject(displayFieldLst);
85                  jsonGen.writeStringField('Id',(string)record.get('Id'));
86                  jsonGen.writeStringField('Label',mapofObjectAPILabel.get(string.valueof(records.getSObjectType())));
87                  jsonGen.writeStringField('ICONURL',mapofObjectIconURL.get(string.valueof(records.getSObjectType())));
88                  jsonGen.writeStringField('ICONName',mapofObjectIconName.get(string.valueof(records.getSObjectType())));
89                  for(String fld : mapofobjFields.get(string.valueof(record.getSObjectType()))){
90                      if(record.get(fld) != null){
91                          jsonGen.writeStringField(fld, String.valueof(record.get(fld)));
92                      }else{
93                          jsonGen.writeStringField(fld,'');
94                      }
95                      for(Integer i=0;i<displayFieldLst.size();i++){
96                          if(fld==displayFieldLst[i]){
97                              string fieldValue = String.valueof(record.get(fld))==NULL ? '' : String.valueof(record.get(fld));
98                              jsonGen.writeStringField('FIELD'+i,fieldValue);
99                          }
100                     }
101                 }
102                 jsonGen.writeStringField('Type',string.valueof(record.getSObjectType()));
103                 jsonGen.writeEndObject();
104             }
105             jsonGen.writeEndArray();
106         }
107         jsonGen.writeEndObject();
108         String jsonData = jsonGen.getAsString();
109         return jsonData;
110     }
```

4. Output :



Code

Access component [here]

References

1. SLDS Custom Lookup

2. JSON Generator in Apex