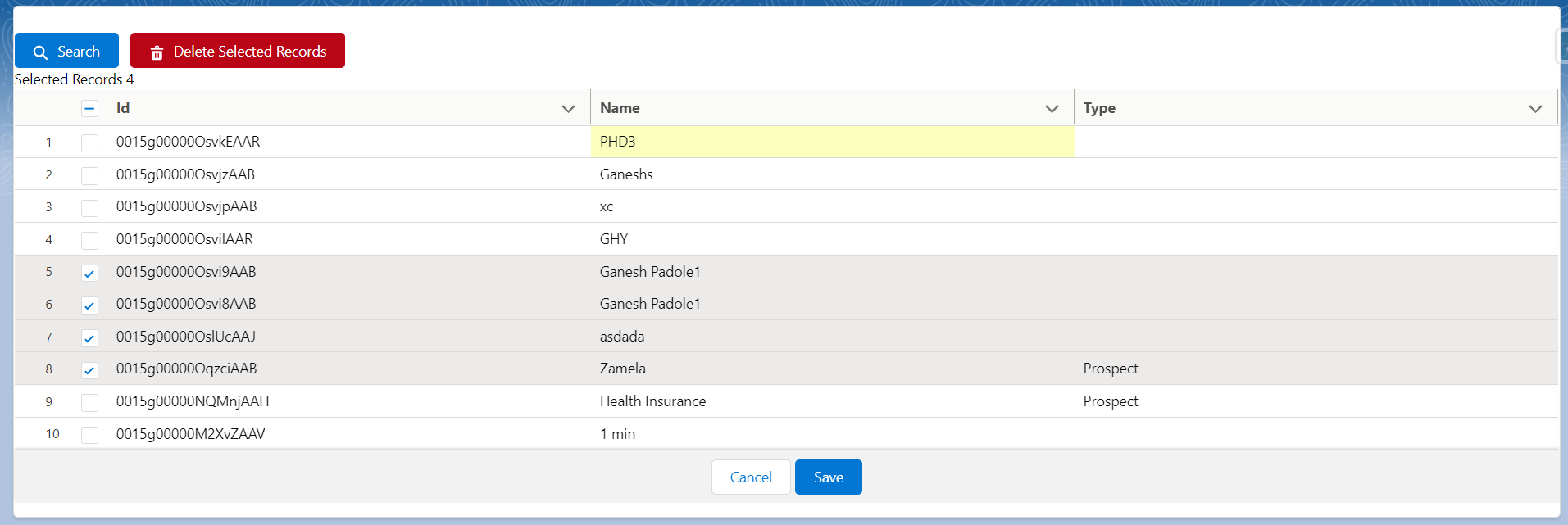
60. Data Table Handling - 28 June 2022

Today we will learn:

1] Data Table Syntax

2] Handling Table Rows Selection

3] Delete Selected Rows



1] Data Table Syntax

**A] HTML FILE**

<lightning-datatable

key-field="Id"

data={accList}

columns={columns}

draft-values={draftValues}

onrowselection={selectedRecordsHandler}

onsave={saveHandler} >

</lightning-datatable>

B] JS Controller Data Table Columns :

const columns = [

{ label: 'Id', fieldName: 'Id', editable: true },

{ label: 'Name', fieldName: 'Name', editable: true },

{ label: 'Type', fieldName: 'Type', editable: true }

];

JS Controller

C] draftValues=[];

    columns = columns;

D] JS Controller Method:

selectedRecordsHandler(event){

const selectedRows = event.detail.selectedRows;

console.log("Selected Rows = "+selectedRows)

this.selectedRecordsCount = event.detail.selectedRows.length;

let recordsSets = new Set();

// getting selected record id

for (let i = 0; i < selectedRows.length; i++) {

recordsSets.add(selectedRows[i].Id);

}

// coverting to array

this.selectedRecords = Array.from(recordsSets);

}

Java Script Data Types

1] const 2] let 3] var

<https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/>

<https://www.geeksforgeeks.org/difference-between-var-let-and-const-keywords-in-javascript/>

<https://www.w3schools.com/js/js_let.asp>

https://www.developer.com/languages/javascript/javascript-var-let-const-variable-declaration/

<!-- sldsValidatorIgnore -->

<template>

    <lightning-card title="Applicant Record ">

        Selected Records = {selectedRecordsCount} <br />

      <lightning-button variant="destructive" icon-name="utility:delete" label="Delete Selected Records" onclick={deleteSelectedRecordsHandler} disabled={disableDeleteButton}></lightning-button>

        <lightning-input  type="search" data-formfield="applicantFirstName" onkeyup={searchApplicantHandler} label="Enter First Name"></lightning-input> <br />

     <template if:true={showTable}>

                <lightning-datatable

                key-field="Id"

                data={applicantList}

                columns={columns}

                draft-values={draftValues}

                onrowselection={selectedRecordsHandler}

                onsave={saveHandler}   >

            </lightning-datatable>

    </template>

    </lightning-card>

</template>

import { LightningElement } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent' ;

import searchApplicantByAlphabets from '@salesforce/apex/ApplicantProvider.searchApplicantByAlphabets';

import deleteSelectedApplicantRecords from  '@salesforce/apex/ApplicantProvider.deleteSelectedApplicantRecords';

export default class ApplicantRecordsDateRange extends LightningElement {

    firstName;

    applicantList;

    draftValues=[];

    selectedRecordsCount=0;

    selectedRecords=0;

    disableDeleteButton = true

    showTable = false;

    columns = [

    { label: 'Id', fieldName: 'Name', editable: true },

    { label: 'First Name', fieldName: 'First\_Name\_\_c', editable: true },

    { label: 'Last Name', fieldName: 'Last\_Name\_\_c', editable: true },

    { label: 'Created Date', fieldName: 'CreatedDate', editable: true }

  ];

   searchApplicantHandler(){

        this.firstName = this.template.querySelector('lightning-input[data-formfield="applicantFirstName"]').value;

        console.log(this.firstName);

        searchApplicantByAlphabets({ firstName: this.firstName})

        .then((result)=>{

            console.log(JSON.stringify(result));

            this.applicantList = result;

            if(this.applicantList.length > 0){

                    this.showTable = true;

            }

            else{

                    this.showTable = false;

            }

        })

        .catch((error) =>{

            console.log(JSON.stringify(error));

        });

    }

    selectedRecordsHandler(event){

        const selectedRows  =   event.detail.selectedRows;//2

        console.log("Selected Rows = "+selectedRows)

        this.selectedRecordsCount = event.detail.selectedRows.length;

        if(this.selectedRecordsCount > 0){

                this.disableDeleteButton = false;

        }

        else{

            this.disableDeleteButton = true;

        }

        let recordsSets = new Set();

        // getting selected record id

        for (let i = 0; i < selectedRows.length; i++) {

            recordsSets.add(selectedRows[i].Id);

        }

        // coverting to array

        this.selectedRecords = Array.from(recordsSets);

      }

      deleteSelectedRecordsHandler(){

        deleteSelectedApplicantRecords({appIdList :  this.selectedRecords, firstName : this.firstName })

        .then((result)=>{

                console.log('Result = '+result);

                 this.applicantList = result;

                console.log('Ramaining Record '+JSON.stringify(result));

                this.showToastMessage('Records deleted successully' , 'success');

        })

        .catch((error)=>{

            console.log('Error = '+error);

        });

      }

      showToastMessage(myMessage, variantName) {

        const evt = new ShowToastEvent({

            title: 'Message',

            message: myMessage,

            variant: variantName,

            mode: 'dismissable'

        });

        this.dispatchEvent(evt);

    }

}

public with sharing class ApplicantProvider {

    @AuraEnabled

    public static string createNewApplicantRecord(Applicant\_\_c objApplicant){

        try {

            if(objApplicant != null ){

                    insert objApplicant;

                    return 'Applicant Record Created Successfully...!!!';

            }

            else{

                    return 'Locha, something went wrong...!!!';

            }

        } catch (Exception e) {

            throw new AuraHandledException(e.getMessage());

        }

    }

   @AuraEnabled

   public static Applicant\_\_c searchApplicantRecord(Applicant\_\_c objApp){

    try {

            if(objApp != null){

             return  ([select Id, First\_Name\_\_c, Last\_Name\_\_c, Gender\_\_c from Applicant\_\_c where Name=: objApp.Name LIMIT 1] );

            }

            else{

                return null;

            }

       } catch (Exception e) {

           throw new AuraHandledException(e.getMessage());

       }

   }

   @AuraEnabled

   public static List<Applicant\_\_c> searchApplicantDateRange(Date fromDate,Date toDate){

    try {

        if(fromDate!=null && toDate !=null){

            return [select Id,Name, First\_Name\_\_c, CreatedDate from Applicant\_\_c where CreatedDate>=:fromDate and CreatedDate <=: toDate];

        }

        else{

            return null;

        }

    } catch (Exception e) {

        throw new AuraHandledException(e.getMessage());

    }

   }

   @AuraEnabled

   public static List<Applicant\_\_c> searchApplicantByAlphabets(String firstName){

    try {

        if( !String.isBlank(firstName)){

        return [select Id, Name, First\_Name\_\_c from Applicant\_\_c where First\_Name\_\_c like : firstName+'%' ];

        }

        else{

            return null;

        }

    } catch (Exception e) {

        throw new AuraHandledException(e.getMessage());

    }

   }

   @AuraEnabled

   public static List<Applicant\_\_c> deleteSelectedApplicantRecords(List<Id> appIdList, String firstName){

    try {

        if( !appIdList.isEmpty()){

          Database.delete([select id from Applicant\_\_c where Id IN : appIdList] , false);

          return [select Id, Name, First\_Name\_\_c, Last\_Name\_\_c, CreatedDate from Applicant\_\_c where First\_Name\_\_c like : firstName+'%'];

        }

        else{

            return null;

        }

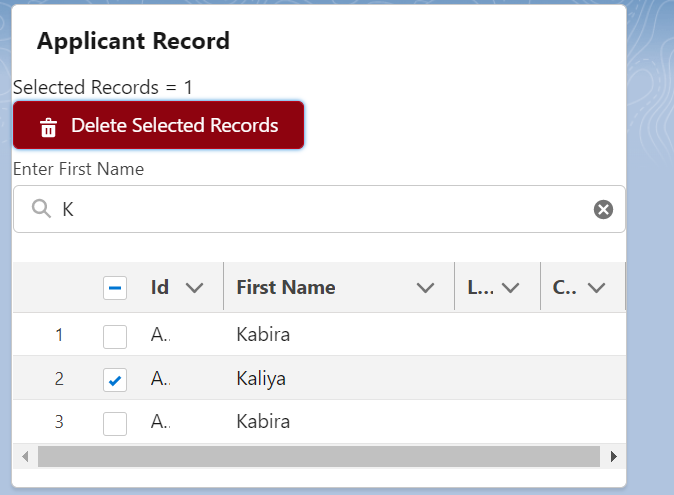
    } catch (Exception e) {

        throw new AuraHandledException(e.getMessage());

    }

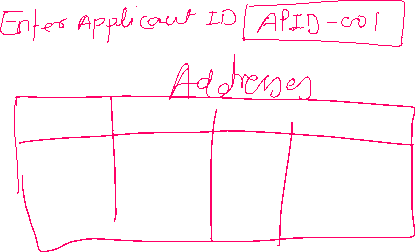
   }

}



Assignment:

User Story 1] Search Applicant ID and Display related Address in a Table. Delete Selected Addresses.



**Search, How to refresh the page / component.**

**Show Latest 10 Account in a Table.**