How to use refreshApex to refresh the list in LWC

In this post , we will see the how to refresh the list in lwc using refreshApex concept.

List of topics covered in this post as below.

- Display the contacts related to selected Account with sortable column & row actions
- Adding new button on top of the account to create new contact
- if user creating the new contact using new button ,the contact list will update automatically.
- if user deleting the any contact using row actions,the contact list will update automatically.

In this logic am using using apex class to create ,delete & return the existing contacts.

To refresh the list , have added the import statement refreshApex in lwc js . if user creating/deleting the contact the cache data becomes stale. so after the creation/deletion method just call the refreshApex to query the server for updated data and refresh the cache.

Apex class :

```
public with sharing class ContactUtilities{

    //Method to return the list of contatcs based on selected account
    @AuraEnabled(Cacheable = true)
    public static List<Contact> getContacts(Id sourceAccount){
        return [SELECT Id, Name, Account.Name, LastName, FirstName, Email, Phone, MobilePhone
                From Contact where AccountId = :sourceAccount];
    }

    //Method to delete the selected contacts
    @AuraEnabled
    public static void deleteContacts(list<Id> deleteContactIds){
        list<Contact> listContactToDelete = new list<Contact>();
        for (Id idContact : deleteContactIds){
            listContactToDelete.add(new Contact(Id = idContact));
        }
        if (!listContactToDelete.isEmpty()){
            delete listContactToDelete;
        }
    }

    //Method to create contact
    @AuraEnabled
    public static void addContact(string firstName, string lastName, String email,id acntId){
        Contact con = new Contact();
        con.FirstName = firstName;
        con.LastName = lastName;
        con.Email = email;
        con.AccountId = acntId;
        try{
            insert con;
        } catch (Exception ex){
            throw new AuraHandledException(ex.getMessage());
        }
    }
}
```

Lightning Web Component

conatctListViewWithRefreshApex.html

```
<template>
    <lightning-card class="slds-card slds-card_boundary related_list_card_border_top" title="ContactListView">

        <!-- New button -->
        <div slot="actions">
            <lightning-button variant="neutral" label="New" type="text" onclick={createNewRecord}
                class="slds-m-right_small">
            </lightning-button>
        </div>

        <!--- Contact Table-->
        <div style="width: auto;">
```

```
        <template if:true={emptyList}>
            <lightning-datatable data={data} columns={columns} key-field="id" show-row-number-column
                hide-checkbox-column="true" onrowaction={handleRowActions} sorted-by={sortBy}
                sorted-direction={sortDirection} onsort={handleSortdata}></lightning-datatable>
        </template>
        <template if:false={emptyList}>
            There is no contacts
        </template>
    </div> <br />

    <!-- Spinner -->
    <div if:true={showLoadingSpinner}>
        <lightning-spinner alternative-text="Loading" size="large"></lightning-spinner>
    </div>

    <!-- Detail view modal -->
    <template if:true={ShowModal}>
        <section role="dialog" tabindex="-1" aria-labelledby="modal-heading-01" aria-modal="true"
            aria-describedby="modal-content-id-1" class="slds-modal slds-fade-in-open">
            <div class="slds-modal__container">

                <!-- modal header -->
                <header class="slds-modal__header">
                    <button class="slds-button slds-button_icon slds-modal__close slds-button_icon-inverse"
                        title="Close" onclick={closeModal}>
                        <lightning-icon icon-name="utility:close" alternative-text="close" variant="inverse"
                            size="small"></lightning-icon>
                    </button>
                    <h2 id="modal-heading-02" class="slds-text-heading_medium slds-hyphenate modalHeading"
                        if:true={editRecord}> Update Contact</h2>
                    <h2 id="modal-heading-0" class="slds-text-heading_medium slds-hyphenate modalHeading"
                        if:true={newRecord}>New Contact</h2>
                </header>

                <!-- showing record edit form -->
                <div if:true={isEditForm} class="slds-theme_default">
                    <lightning-record-edit-form layout-type="Full" object-api-name="Contact" record-id={currentRecordId}>
                        <lightning-messages></lightning-messages>
                        <div if:true={error}>
                            {error}
                        </div>
                        <div class="slds-col slds-size_1-of-1">
                            <lightning-input-field field-name="FirstName" onchange={firstNameChange}>
                            </lightning-input-field>
                        </div>
                        <div class="slds-col slds-size_1-of-1">
                            <lightning-input-field field-name="LastName" onchange={lastNameChange}>
                            </lightning-input-field>
                        </div>
                        <div class="slds-col slds-size_1-of-1">
                            <lightning-input-field field-name="Email" onchange={emailChange}>
                            </lightning-input-field>
                        </div>

                        <br />
                        <ul class="slds-button-group-row slds-grid slds-grid_align-center">
                            <li class="slds-button-group-item">
                                <lightning-button class="slds-m-top_small" variant="brand" name="update"
                                    label="save" onclick={handleSubmit}></lightning-button>
                            </li>
                            <li class="slds-button-group-item">
                                <lightning-button class="slds-m-top_small" variant="brand" type="text"
                                    label="Cancel" onclick={closeModal}></lightning-button>
                            </li>
                        </ul>
                    </lightning-record-edit-form><br/>
                    <div></div>
                </div>
            </div>
        </section>

        <div class="slds-backdrop slds-backdrop_open"></div>
    </template>
</lightning-card>
</template>
```

conatctListViewWithRefreshApex.js

```
import { LightningElement, track, wire, api } from 'lwc';
```

```
//Import Apex methods
import getContatcs from '@salesforce/apex/ContactUtilities.getContacts';
import deletecontact from '@salesforce/apex/ContactUtilities.deleteContacts';
import createContact from '@salesforce/apex/ContactUtilities.addContact';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { refreshApex } from '@salesforce/apex';
import { NavigationMixin } from 'lightning/navigation';

// row actions
const actions = [
    { label: 'Delete', name: 'delete' }
];

// datatable columns with row actions
const columns = [
    { label: 'First Name', fieldName: 'FirstName', type: "string", sortable: true },
    { label: 'Last Name', fieldName: 'LastName', type: "string", sortable: true },
    { label: 'Email', fieldName: 'Email', type: "email", sortable: true },
    {
        type: 'action',
        typeAttributes: {
            rowActions: actions,
            menuAlignment: 'right'
        }
    }
];

export default class ConatctListViewWithRefreshApex extends NavigationMixin(LightningElement) {

    // reactive variable
    @api recordId;
    @track data;
    @track columns = columns;
    @track record = [];
    @track ShowModal = false;
    @track currentRecordId;
    @track isEditForm = false;
    @track showLoadingSpinner = false;
    @track newRecord = false;
    @track editRecord = false;
    @track viewRecord = false;
    @track sortBy;
    @track sortDirection;
    @track emptyList = false;
    @track error;
    @track firstName;
    @track lastName;
    @track email;


    // non-reactive variables
    refreshTable;

    // retrieving the data using wire service
    @wire(getContatcs, { sourceAccount: '$recordId' })
    relations(result) {
        this.refreshTable = result;
        if (result.data) {
            this.data = result.data;
            this.emptyList = true;
        }
    }

    handleSortdata(event) {
        // field name
        this.sortBy = event.detail.fieldName;
        // sort direction
        this.sortDirection = event.detail.sortDirection;
        // calling sortdata function to sort the data based on direction and selected field
        this.sortData(event.detail.fieldName, event.detail.sortDirection);
    }

    sortData(fieldname, direction) {
        // serialize the data before calling sort function
        let parseData = JSON.parse(JSON.stringify(this.data));
        // Return the value stored in the field
        let keyValue = (a) => {
            return a[fieldname];
        };
```

```
        // cheking reverse direction
        let isReverse = direction === 'asc' ? 1 : -1;
        // sorting data
        parseData.sort((x, y) => {
            x = keyValue(x) ? keyValue(x) : ''; // handling null values
            y = keyValue(y) ? keyValue(y) : '';

            // sorting values based on direction
            return isReverse * ((x > y) - (y > x));
        });

        // set the sorted data to data table data
        this.data = parseData;

    }

    //To handle the row actions
    handleRowActions(event) {
        let actionName = event.detail.action.name;
        let row = event.detail.row;
        // eslint-disable-next-line default-case
        switch (actionName) {
            case 'delete':
                this.deleteRelations(row);
                break;
        }
    }

    //To create new record
    createNewRecord() {
        // open modal box
        this.currentRecordId = '';
        this.ShowModal = true;
        this.isEditForm = true;
        this.newRecord = true;
    }

    // handleing record edit form submit
    handleSubmit() {
        createContact({ firstName: this.firstName, lastName: this.lastName, email: this.email, acntId: this.recordId })
            .then(result => {
                this.handleSuccess();
            })
            .catch(error => {
                this.error = error.body.message;
            });
    }

    // refreshing the datatable after record edit form success
    handleSuccess() {
        // closing modal
        this.ShowModal = false;
        // showing success message
        if (this.newRecord === true) {
            this.dispatchEvent(new ShowToastEvent({
                message: 'Contact Created sucessfully',
                variant: 'success'
            }));

        }
        if (this.newRecord === false) {
            this.dispatchEvent(new ShowToastEvent({
                message: 'Contact updated sucessfully',
                variant: 'success'
            }));
        }
        return refreshApex(this.refreshTable);
    }

    //To delete the selected relations
    deleteRelations(currentRow) {
        let currentRecord = [];
        currentRecord.push(currentRow.Id);
        this.showLoadingSpinner = true;

        // calling apex class method to delete the selected contact
        deletecontact({ deleteContactIds: currentRecord })
            .then(result => {
                this.showLoadingSpinner = false;
```

```
            // showing success message
            this.dispatchEvent(new ShowToastEvent({
                message: 'Contatact Deleted sucessfully',
                variant: 'success'
            }));

            // refreshing table data using refresh apex
            return refreshApex(this.refreshTable);

        })
        .catch(error => {
            this.dispatchEvent(new ShowToastEvent({
                message: error.message,
                variant: 'error'
            }));
        });
}


//Event to track the First Name field
firstNameChange(event) {
    this.firstName = event.target.value;
    this.error = '';
}

//Event to track the last Name field
lastNameChange(event) {
    this.lastName = event.target.value;
    this.error = '';
}


//Event to track the Email field
emailChange(event) {
    this.email = event.target.value;
    this.error = '';
}

// closing modal box
closeModal() {
    this.ShowModal = false;
    this.newRecord = false;
    this.editRecord = false;
    this.viewRecord = false;
    this.error = '';
}

}
```

conatctListViewWithRefreshApex.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>48.0</apiVersion>
    <isExposed>True</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
    <targetConfigs>
      <targetConfig targets="lightning__RecordPage">
          <objects>
              <object>Account</object>
          </objects>
      </targetConfig>
  </targetConfigs>
</LightningComponentBundle>
```

GitHub:
https://github.com/rcsaravananmkd/Saravanan/tree/master/LWC/conatctListViewWithRefreshApex

Demo :