

Example on wire

Create a class first

```
public with sharing class contactlist {  
    @AuraEnabled(Cacheable=true)  
    public static list<contact> getcont()  
    {  
        list<contact> conlist=[select firstname from contact limit 10];  
        return conlist;  
    }  
}
```

Create Lwc Component

```
<template>
```

```
    <template for:each={conlist.data} for:item="contact">
```

```
        <lightning-card key={contact.FirstName} icon-  
name="standard:contact">
```

```
    <p>
```

```
        {contact.FirstName}
```

```
    </p>
```

```
    </lightning-card>
```

```
    </template>
```

```
</template>
```

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

```
import getcont from '@salesforce/apex/contactlist.getcont';
```

```
export default class Contactlistwire extends LightningElement {
    @track conlist;
    @wire(getcont) conlist;
}
```

.....

Call into aura application

```
<aura:application extends="force:slds">
    <c:contactlistwire></c:contactlistwire>
```

```
</aura:application>
```

.....

In case user want to add an error also

```
import { LightningElement, track, wire } from 'lwc';
import getcont from '@salesforce/apex/contactlist.getcont';
```

```
export default class Contactlistwire extends LightningElement {
    @track conlist;
    //@wire(getcont) conlist;
    @wire(getcont) contactl({data,error}){

        if(data){
            this.conlist=data;
        }
        else if(error){
            console.log('error #' + error);
        }
    }

    //@wire executed before the connectedcallback and render method
    //when component is ready after that wire works
```

```

}

<template>

    <template for:each={conlist} for:item="contact"> //from here
removedata

        <lightning-card key={contact.FirstName} icon-
name="standard:contact">

<p>

        {contact.FirstName}

</p>

    </lightning-card>

</template>

</template>

```

.....

In case if user want data on a button call then user can use imperatively method

Calling an apex method imperatively

When: Suppose you want to call apex method on click of a button.

Steps:

1. Make the apex method @AuraEnabled.
2. Import the apex method into javascript file of your component using:
import apexMethod from '@salesforce/apex/namespace.Classname.apexMethod';
3. Make Call to the method

```

apexMethod()
    .then(result => {
        this.resultVar = result;
    })
    .catch(error => {
        this.error = error;
    });

```

Lightning Web Components:@Wire and Imperative Apex Server Calls in LWC

13,354 views • Premiered on 16 Jan 2020

210 DISLIKE SHARE CLIP SAVE

SALESFORCE PREDATOR

Chat Replay has been disabled for this Premiere.

All JavaScript C# Computer Application

LIGHTNING WEB COMPONENT CRASH COURSE
SFDC Facts Academy
169K views • 2 years ago

Lightning Web Components(LWC)
SALESFORCE PREDATOR

Salesforce developer interview questions and answers || 2022...
Ashfaq Mohammed
1.7K views • 1 month ago

Lightning Web Component-04 ("wire" decorator and call apex...
Salesforce Techbook
9.7K views • 2 years ago

Lightning Web Components Best practices
Salesforce Apex Hours
1.9K views • 2 years ago

Salesforce Trainin...mp4

Type here to search

39°C Haze 15:48 10-05-2022

See an example

```
<template>

  <template for:each={conlist} for:item="contact">

    <lightning-card key={contact.FirstName} icon-
name="standard:contact">

      <p>

        {contact.FirstName}

      </p>

    </lightning-card>

  </template>

  <lightning-button label="getcontacts"
onclick={getconts}></lightning-button>

</template>
```

In js

```
/*@wire(getcont) contact1({data,error}){

  if(data){
    this.conlist=data;
  }
  else if(error){
    console.log('error #' + error);
  }
}
```

```

//@wire executed before the connectedcallback and render method
//when component is ready after that wire works
*/
//using imperative call
getconts() {
    getcont().then(result=>{this.conlist=result;
    })
    .catch(error=>{

        console.log('error'+error);
    })
}

```

..... .

Write an Apex method for dynamic parameter

Add parameter of boolean with name showcontact in class

Then add this code in js

```

import { LightningElement, track, wire } from 'lwc';
import getcont from '@salesforce/apex/contactlist.getcont';

export default class Contactlistwire extends LightningElement {
    @track conlist;
    //@wire(getcont) conlist;
    @track showcontact=false;
    @wire(getcont, {flag: '$showcontact'}) contactl({data,error}){

        if(data){
            this.conlist=data;
        }
        else if(error){
            console.log('error #' + error);
        }
    }
}

```

```

    }
}
handlechange(event) {
    this.showcontact=event.target.checked;

}

}

```

In html

```

<template>
<template if:true={showcontact}>

    <template for:each={conlist} for:item="contact">

        <lightning-card key={contact.FirstName} icon-
name="standard:contact">
<p>
    {contact.FirstName}

</p>
    </lightning-card>
    </template>
</template>

    <lightning-input type="checkbox" label="show contacts"
checked={showcontact} onchange={handlechange}></lightning-input>

</template>

```

..... • •

.....

Create messagechannel folder in
Sample MessageChannel.messageChannel-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningMessageChannel xmlns="https://soap.sforce.com/2006/04/metadata">
  <description>This is a sample Lightning Message Channel.</description>
  <isExposed>true</isExposed>
  <lightningMessageFields>
    <description>Variable 1</description>
    <fieldName>variable1</fieldName>
  </lightningMessageFields>
  <masterLabel>Sample Message Channel</masterLabel>
</LightningMessageChannel>
```