Check your orgs readiness for lightning experience

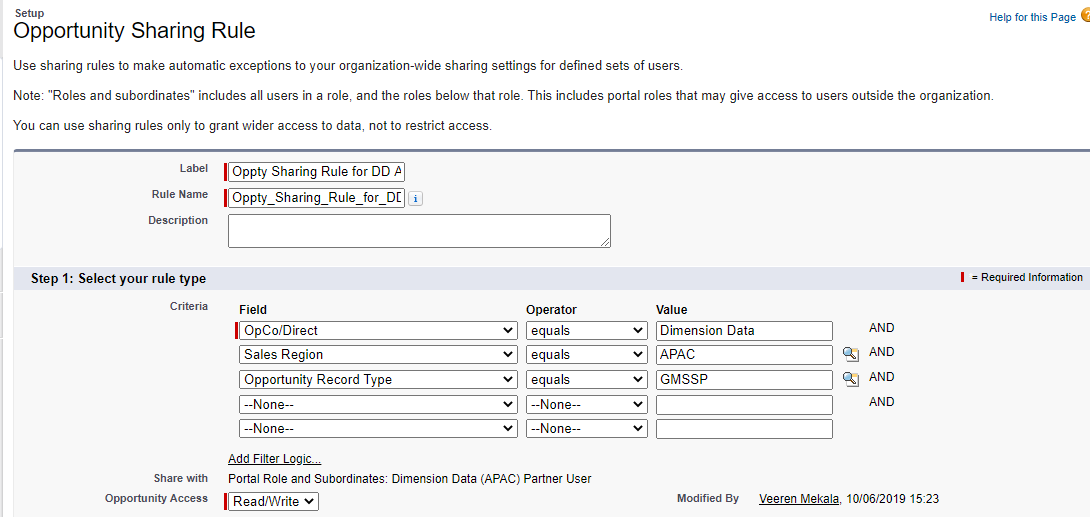
Lightning Experience Configuration Converter to convert javascript, Actions and button and Visualforcepage in lightning

* ransition.
* [**Discover Phase: Start Your Lightning Experience Transition**](https://help.salesforce.com/articleView?id=lex_phases_discover.htm&type=5)  
  Kick off your Lightning Experience transition with a discovery period. Learn about the benefits and ROI that your company can get from moving to Lightning Experience and evaluate your production org's readiness for the new interface. Then plan your rollout and change management strategies and align with stakeholders on your transition goals.
* [**Roll Out Phase: Get Ready to Launch Lightning Experience**](https://help.salesforce.com/articleView?id=lex_phases_roll_out.htm&type=5)  
  With your rollout plan in place and your stakeholders on board, you’re ready to start preparing your org—and your users—for Lightning Experience. This is the time to kick off your change management strategy. And time to get busy on release development activities, including implementing Lightning Experience features and addressing the priority issues noted in your Readiness Report. The last piece is going live by giving users access the new interface!
* [**Optimize Phase: Make Lightning Experience Stick**](https://help.salesforce.com/articleView?id=lex_phases_optimize.htm&type=5)  
  The transition project doesn't end when you give users access to Lightning Experience. Busy people aren't always eager for change, so keep an eye on adoption rates and encourage users to stay in the new interface. This is also the time to iterate on your rollout plan by moving over more users, adding new features and use cases, and looking for ways to improve your implementation. And when the day comes to turn off access to Salesforce Classic, you can call your transition a success!
* [**Find Your Way Around Lightning Experience**](https://help.salesforce.com/articleView?id=lex_find_your_way_around.htm&type=5)  
  Feeling a little discombobulated by all the Lightning Experience newness? Perfectly understandable. Diving into a redesigned app can be disorienting—especially when you’re able to work in the original interface with your eyes closed. To hone your Lightning Experience chops, let’s look at where some of the most common Salesforce features landed in the new interface.
* [**Lightning Experience Considerations**](https://help.salesforce.com/articleView?id=lex_considerations_intro.htm&type=5)  
  Lightning Experience re-envisions Salesforce and as a result, there are some classic features that have been implemented differently or don’t have a place in the new interface. And there are some classic features that are on the roadmap for an upcoming release. Review these topics to see if any functionality that’s critical to your business has changed significantly in Lightning Experience, or is limited or unavailable in this release. This information can help you train users about what to expect when you roll out Lightning Experience.

Security control – sharing settings, field accessbility

Sharing Settings - access to data for users - has section OWD ( grant access using hierarchy, default access) and sharing rules for all the objects.

Most of the objects like accounts, contacts have public read/write except opportunity and quote. shareing rules defined to give access to specific sets of opps.



Visualforce email template

Apex:outputtext to write email body, value = {!account.name}

Set rendered = ‘false’ if you want to hide

!$Label.ApttusEmailServiceAddress for reply to address

Apex:attribute is parameter passed from parent form or email template

Apex:Outputpanel to show a table conditionally,

<apex:outputPanel rendered="{!IF(contains(approvalRequest.Approval\_Request\_For\_\_c,'Wholesale'),TRUE,FALSE)}">

<messaging:emailTemplate subject="NSTAR : '{!relatedTo.Apttus\_Approval\_\_Step\_Name\_\_c}' - Review and Approve New Request" recipientType="User" relatedToType="Apttus\_Approval\_\_Approval\_Request\_\_c"

replyTo="{!$Label.ApttusEmailServiceAddressNew}"><messaging:htmlEmailBody >

<apex:outputText value="{!relatedTo.Apttus\_Approval\_\_Step\_Name\_\_c}" rendered="false" />

Dear {!recipient.Name}:

<br/><br/><br/>

Approval Request for '{!relatedTo.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Apttus\_Proposal\_\_Opportunity\_\_r.Name}', request type '{!relatedTo.Apttus\_Approval\_\_Step\_Name\_\_c}' has been submitted.

<br/><br/>

Quote Number : {!relatedTo.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Name}

<br/>

2. Make approval via this link <a href="{!$Label.My\_Approval\_Base\_URL}?id={!relatedTo.Apttus\_Approval\_\_ObjectRefId\_\_c}">My Approvals</a><br/>

<c:NTT\_OpCoApprovalNotificationComponent approvalRequestSO="{!relatedTo}" mailReceiver="{!recipient}" emailTriggeringEventStr="aprSubmission" />

</messaging:htmlEmailBody>

</messaging:emailTemplate>

<apex:component controller="NTT\_OpCoApprovalNotificationController" access="global">

<apex:attribute name="approvalRequestSO"

assignTo="{!approvalRequest}"

type="Apttus\_Approval\_\_Approval\_Request\_\_c"

description="To retain Approval Request." />

<apex:attribute name="mailReceiver"

assignTo="{!recipient}"

type="User"

description="To retain mail recipient." />

<apex:outputPanel rendered="{!IF(contains(approvalRequest.Approval\_Request\_For\_\_c,'Wholesale'),TRUE,FALSE)}">

<table width="80%" border="1" style="border:1px solid black;border-collapse:collapse;" >

<thead>

<tr colspan="3">

<b> <u>Wholesale Information</u> </b> <br/><br/>

</tr>

<tr>

<th>

Total Contract Value Wholesale

</th>

<th>

Total Subscription Wholesale Price

</th>

<th>

Total One-time Wholesale Price

</th>

</tr>

</thead>

<tbody>

<tr>

<td>

<apex:outputText value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.CurrencyIsoCode} {0, number, #,##0.00}">

<apex:param value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Total\_Contract\_Value\_Wholesale\_\_c}" />

</apex:outputText>

</td>

<td>

<apex:outputText value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.CurrencyIsoCode} {0, number, #,##0.00}">

<apex:param value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Total\_Subscription\_Wholesale\_Price\_\_c}" />

</apex:outputText>

</td>

<td>

<apex:outputText value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.CurrencyIsoCode} {0, number, #,##0.00}">

<apex:param value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Total\_One\_Time\_Wholesale\_Price\_\_c}" />

</apex:outputText>

</td>

</tr>

</tbody>

</table>

</apex:outputPanel>

<apex:outputLabel value="Wholesale Discount % = " style="font-weight:bold;" rendered="{!IF(approvalRequest.Approval\_Request\_For\_\_c == 'Wholesale Price Discount',TRUE,FALSE)}" />

<apex:outputText value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Wholesale\_Disc\_Uplift\_\_c}" rendered="{!IF(approvalRequest.Approval\_Request\_For\_\_c == 'Wholesale Price Discount', TRUE, FALSE)}" />

<apex:outputText value="{0,date,dd-MMM-yyyy}"><apex:param value="{!approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_r.Apttus\_Proposal\_\_Opportunity\_\_r.Contract\_Start\_Date\_\_c}"/>

</apex:outputPanel>

</apex:component>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Name : NTT\_OpCoApprovalNotificationController  
 \* Description : Controller class for VF Component created for OpCo Approval Process.  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
global with sharing class NTT\_OpCoApprovalNotificationController {  
      
    public static Apttus\_Approval\_\_Approval\_Request\_\_c approvalRequest {get; set;}  
    public static String emailTriggeringEvent {get; set;}  
    public static User recipient {get; set;}  
    public static String ApprovalInfo{   
        get{   
             
             Apttus\_Proposal\_\_Proposal\_\_c[] proposals = [Select Approval\_Notes\_\_c from Apttus\_Proposal\_\_Proposal\_\_c where id = :approvalRequest.Apttus\_CQApprov\_\_CartId\_\_r.Apttus\_QPConfig\_\_Proposald\_\_C];                               
             if (proposals.size()>0)  
                ApprovalInfo = proposals[0].Approval\_Notes\_\_c;  
            return ApprovalInfo;  
        }   
        set;  
         
    }  
}

@isTest  
public class NTT\_OpCoApprovalNotificationCtlrTest {  
      
    public static testMethod void testController() {  
          
        Apttus\_Approval\_\_Approval\_Request\_\_c request= new Apttus\_Approval\_\_Approval\_Request\_\_c();  
        User recipient = new User();  
        string approvalinfo;  
         
    NTT\_OpCoApprovalNotificationController.approvalRequest = request;  
    NTT\_OpCoApprovalNotificationController.recipient = recipient;  
    NTT\_OpCoApprovalNotificationController.emailTriggeringEvent = 'Test';         
         
    approvalinfo = NTT\_OpCoApprovalNotificationController.ApprovalInfo;  
          
    System.debug('' + request);  
    System.debug('' + recipient);  
    }  
}

Visualforce page to upload in csv. To display Visualforce page data in a Microsoft Excel spreadsheet, use the contentType attribute on the <apex:page> tag, and specify a value of application/vnd.ms-excel.

Convert a page to PDF by changing the <apex:page> tag.

[view source](https://developer.salesforce.com/docs/#viewSource)

[print](https://developer.salesforce.com/docs/#printSource)[?](https://developer.salesforce.com/docs/#about)

|  |  |
| --- | --- |
|  | <apex:page renderAs="pdf"> |

<apex:page controller="RegionUploadsController" readonly="true" contentType="application/vnd.ms-excel#RegionUploadFile.csv" cache="true">Client Name,Sales Country,Quote date Expiry date,Estimated Start Date,Quote Ref,Approval Stage,Contract Type,Service,Mfr Part Num,Service Desk,Service Package,Delivery Entity,Description,Cost Type,Cost Currency,Exc to Deal Currency,Deal Currency,Contract Period,Selling Frequency,Unit Description,Unit Count,Unit Quantity,Unit Cost, Unit Price,Total Cost , Total Price, Margin %, One time cost per line, One time price per line, Monthy cost per line,Monthy price per line,Contract life cost per line,Contract life Price Perline,Total One Time Cost,Total One Time Price,Total One Time Margin,Total One Time Margin %,Total Monthly Cost,Total Monthly Price,Total Monthly Margin,Total Monthly Margin %,Total Contract Life Cost,Total Contract Life Price,Total Contract Life Margin,Total Contract Life Margin %

<apex:repeat value="{!proposalLineItems}" var="pli">

<apex:variable var="costCurrency" value="{!If(pli.Apttus\_QPConfig\_\_DerivedFromId\_\_r.Apttus\_Config2\_\_AttributeValueId\_\_r.Language\_Support\_\_c=='Group Service Desk', 'USD', pli.CurrencyIsoCode)}" />

{!pli.Apttus\_Proposal\_\_Proposal\_\_r.End\_Customer\_Text\_\_c},{!pli.Apttus\_Proposal\_\_Proposal\_\_r.Sales\_Organization\_\_c},

<apex:outputText value="{0,date,MM'/'dd'/'yyyy}">

<apex:param value="{!pli.Apttus\_QPConfig\_\_EndDate\_\_c}" />

</apex:outputText>,<apex:outputText value="{0,date,MM'/'dd'/'yyyy}">

<apex:param value="{!pli.Apttus\_QPConfig\_\_StartDate\_\_c}" />

</apex:outputText>,{!pli.Apttus\_Proposal\_\_Proposal\_\_r.name},{!pli.Apttus\_Proposal\_\_Proposal\_\_r.Apttus\_Proposal\_\_Approval\_Stage\_\_c},{!pli.Apttus\_Proposal\_\_Proposal\_\_r.Opportunity\_Type\_\_c},{!pli.Level\_2\_\_c},{!pli.CP\_Mfr\_Part\_Number\_\_c},{!pli.Apttus\_QPConfig\_\_DerivedFromId\_\_r.Apttus\_Config2\_\_AttributeValueId\_\_r.Language\_Support\_\_c},{!pli.Apttus\_QPConfig\_\_AttributeValueId\_\_r.Managed\_Network\_Service\_Level\_\_c},,{!pli.Apttus\_QPConfig\_\_OptionId\_\_r.name} {!pli.Quantity\_in\_Template\_\_c} {!pli.Apttus\_QPConfig\_\_SellingUom\_\_c},{!pli.CP\_Cost\_Type\_\_c},{!costCurrency},{!mapCurrencyCodeToUsdValue[pli.CurrencyIsoCode+costCurrency]},{!pli.CurrencyIsoCode},{!Round(pli.Apttus\_QPConfig\_\_SellingTerm\_\_c,0)},{!pli.Apttus\_QPConfig\_\_SellingFrequency\_\_c},{!pli.Apttus\_QPConfig\_\_SellingUom\_\_c},{!pli.Apttus\_QPConfig\_\_PriceListItemId\_\_r.Unit\_Count\_\_c},{!Round(pli.Apttus\_QPConfig\_\_Quantity2\_\_c,0)},{!Round(pli.CP\_UnitCost\_\_c,2)},{!Round(pli.CP\_Deal\_UnitPrice\_\_c,2)},{!pli.CP\_Net\_Cost\_One\_Time\_\_c},{!pli.CP\_Net\_Deal\_Price\_One\_Time\_\_c},{!pli.CP\_Deal\_Margin\_Pct\_\_c},{!pli.CP\_Net\_Cost\_One\_Time\_\_c},{!pli.CP\_Net\_Deal\_Price\_One\_Time\_\_c},{!pli.CP\_Cost\_Per\_Month\_\_c},{!pli.CP\_Deal\_Price\_Per\_Month\_\_c},{!pli.CP\_Net\_Cost\_\_c},{!Round(pli.CP\_Net\_DealPrice\_\_c,2)},{!TotalOneTimeCost},{!TotalOneTimePrice},{!TotalOneTimeMargin},{!Round(TotalOneTimeMarginPercent\*100,0)},{!TotalMonthlyCost},{!TotalMonthlyPrice},{!TotalMonthlyMargin},{!Round(TotalMonthlyMarginPercent\*100,0)},{!TotalContractLifeCost},{!TotalContractLifePrice},{!TotalContractLifeMargin},{!Round(TotalContractLifeMarginPercent\*100,0)}

</apex:repeat>

</apex:page>

public class RegionUploadsController {  
    String id ;  
    public List<Apttus\_Proposal\_\_Proposal\_Line\_Item\_\_c> proposalLineItems {get;set;}  
    Public Decimal TotalOneTimeCost{get;set;}  
    Public Decimal TotalOneTimePrice{get;set;}    
    Public Map<String, Decimal> mapCurrencyCodeToUsdValue {get;set;}   
    public RegionUploadsController() {  
        id = ApexPages.currentPage().getParameters().get('id');  
        TotalOneTimeCost=0;  
        TotalOneTimePrice=0;  
          
        mapCurrencyCodeToUsdValue = new Map<String, Decimal>();  
        run();  
    }  
      
    public PageReference run() {  
          
        proposalLineItems = [SELECT Id,  
                             Apttus\_QPConfig\_\_LineNumber\_\_c,   
                             Apttus\_QPConfig\_\_ItemSequence\_\_c,   
                             Apttus\_QPConfig\_\_LineType\_\_c,   
                             Apttus\_QPConfig\_\_DerivedFromId\_\_r.Apttus\_Config2\_\_AttributeValueId\_\_r.Language\_Support\_\_c // service desk location  
                             FROM Apttus\_Proposal\_\_Proposal\_Line\_Item\_\_c  
                             WHERE Apttus\_Proposal\_\_Proposal\_\_c = :id AND Apttus\_QPConfig\_\_LineType\_\_c = 'option'  
                             Order BY Apttus\_QPConfig\_\_LineNumber\_\_c, Apttus\_QPConfig\_\_ItemSequence\_\_c];  
          
        Map<String, Group\_Currency\_Rate\_\_c> mapCurrencyCodeToRate = new Map<String, Group\_Currency\_Rate\_\_c>();  
        for(Group\_Currency\_Rate\_\_c currencyRate:[SELECT Id, Currency\_Code\_\_c, From\_USD\_rate\_\_c, To\_USD\_rate\_\_c from Group\_Currency\_Rate\_\_c ])  
        {  
            mapCurrencyCodeToRate.put(currencyRate.Currency\_Code\_\_c , currencyRate );  
        }  
          
        for (Apttus\_Proposal\_\_Proposal\_Line\_Item\_\_c lineitem:proposalLineItems){   
            TotalOneTimeCost+=lineitem.CP\_Net\_Cost\_One\_Time\_\_c;  
            TotalOneTimePrice+=lineitem.CP\_Net\_Deal\_Price\_One\_Time\_\_c;

            if(mapCurrencyCodeToRate.containsKey(lineitem.CurrencyIsoCode)){  
                Decimal decRate = lineitem.CurrencyIsoCode == costCurrency ? 1: mapCurrencyCodeToRate.get(lineitem.CurrencyIsoCode).From\_USD\_rate\_\_c;  
                mapCurrencyCodeToUsdValue.put(lineitem.CurrencyIsoCode + costCurrency, decRate);  
            }  
            return null;  
    }  
}

Write test class for controller

   @isTest  
    public static void testRegionUpload(){  
        Apttus\_Proposal\_\_Proposal\_\_c proposal = [SELECT Id from Apttus\_Proposal\_\_Proposal\_\_c limit 1];  
        ApexPages.currentPage().getParameters().put('id', proposal.Id);  
        RegionUploadsController ctrl = new RegionUploadsController();  
          
    }

Asset line item filter

/\*  
This AssetLineItemCallback is being used to add the capability of filtering the Installed Products list based on the Ship-to on the Quote/Proposal Header  
The method being used is : getFilterExpr  
August 10 2018  
\*/  
  
global with sharing class AssetLineItemCallback implements Apttus\_Config2.CustomClass.IAssetLineItemCallback2{  
      
    @TestVisible private String assetSearchFilter = null;  
    @TestVisible private List<String> assetSearchScope = null;  
    @TestVisible private ID priceListId = null;  
    @TestVisible private Id QuoteId=null;  
      
    public AssetLineItemCallback() {  
     }   
       
     global void start(Apttus\_Config2.ProductConfiguration cart,  String assetSearchFilter, List<String> assetSearchScope) {  
         this.assetSearchFilter = assetSearchFilter;  
         this.assetSearchScope = assetSearchScope;  
         //this.priceListId = cart.getConfigSO().PriceListId\_\_c;  
         if (cart != null)  
             QuoteId=cart.getConfigSO().Apttus\_QPConfig\_\_Proposald\_\_c; //Fetching the Quote Id from the CartId param.  
     }  
       
     global String getQueryFilter(ID accountId) {  
       return '';  
     }  
    global List<String> getAssetSearchScope(){   
         return this.assetSearchScope;  
    }  
      
    global String getFilterExpr(Apttus\_Config2.CustomClass.ActionParams params) {  
      
    //Querying the Proposal from the QuoteID to get the Ship to account ID  
    List<Apttus\_Proposal\_\_Proposal\_\_c> propList=[Select Id, Apttus\_QPConfig\_\_ShipToAccountId\_\_c,Apttus\_Proposal\_\_Account\_\_c, Previous\_Order\_Number\_\_c    
                                                 from Apttus\_Proposal\_\_Proposal\_\_c where Id=:QuoteId];  
      
    // If no previous order number, no installed products, leaving this code in case want to display all installed products  
    if (propList[0].Previous\_Order\_Number\_\_c != null)  
     {  
      if(propList!=null && propList.size()>0 && propList[0]!=null)  
      {  
          //Comparing the ShipTo Account of Quote with the Asset Line Item.  
          if(propList[0].Apttus\_QPConfig\_\_ShipToAccountId\_\_c!=null)  
               return 'Apttus\_Config2\_\_ShipToAccountId\_\_c = \'' + propList[0].Apttus\_QPConfig\_\_ShipToAccountId\_\_c + '\' AND Order\_\_c = \'' + propList[0].Previous\_Order\_Number\_\_c +  
                  '\' AND Apttus\_Config2\_\_AccountId\_\_c = \'' + propList[0].Apttus\_Proposal\_\_Account\_\_c+ '\' AND Apttus\_Config2\_\_ChargeType\_\_c!= \'One Time\'';  
          else  
              return 'Apttus\_Config2\_\_ShipToAccountId\_\_c!=null AND Apttus\_Config2\_\_AccountId\_\_c =\'' + propList[0].Apttus\_Proposal\_\_Account\_\_c + '\' AND Order\_\_c = \'' + propList[0].Previous\_Order\_Number\_\_c +  
                  '\' AND Apttus\_Config2\_\_ChargeType\_\_c!= \'One Time\'';  
              
      }          
    }   
       
    if(propList!=null && propList.size()>0 && propList[0]!=null)  
    {  
        //Comparing the ShipTo Account of Quote with the Asset Line Item.  
        if(propList[0].Apttus\_QPConfig\_\_ShipToAccountId\_\_c!=null)  
          return 'Apttus\_Config2\_\_ShipToAccountId\_\_c = \'' + propList[0].Apttus\_QPConfig\_\_ShipToAccountId\_\_c +   
                  '\' AND Apttus\_Config2\_\_AccountId\_\_c = \'' + propList[0].Apttus\_Proposal\_\_Account\_\_c+ '\' AND Apttus\_Config2\_\_ChargeType\_\_c!= \'One Time\'';  
         else  
          return 'Apttus\_Config2\_\_ShipToAccountId\_\_c!=null AND Apttus\_Config2\_\_AccountId\_\_c =\'' + propList[0].Apttus\_Proposal\_\_Account\_\_c +   
                  '\' AND Apttus\_Config2\_\_ChargeType\_\_c!= \'One Time\'';  
    }  
      
    return null;  
      
       
     }  
       
    global void finish() {}  
}

@isTest  
public class AssetLineItemCallbackTest {  
        
    @isTest public static void test1(){  
          
        List<String> sc = null;  
        AssetLineItemCallback AST1 = new AssetLineItemCallback();  
        AST1.assetSearchFilter = null;  
        AST1.assetSearchScope = sc;  
        AST1.priceListId = null;  
        AST1.QuoteId =null;  
      
        Map<String, Schema.RecordTypeInfo> mapRType = Schema.SObjectType.Apttus\_Proposal\_\_Proposal\_\_c.getRecordTypeInfosByName();  
        Schema.RecordTypeInfo pRT = mapRType.get('OpCo');  
        Account A1 = new Account (Name='Test1');  
        Insert A1;    
        Account A2  = new Account (Name='Test2');  
        Insert A2;  
        Apttus\_Proposal\_\_Proposal\_\_c testProposal = new Apttus\_Proposal\_\_Proposal\_\_c(RecordTypeID = pRT.getRecordTypeID());  
        testProposal.Apttus\_QPConfig\_\_ShipToAccountId\_\_c = A2.Id;  
        testProposal.Apttus\_Proposal\_\_Account\_\_c = A1.Id;   
        insert testProposal;     
        Apttus\_Config2\_\_ProductConfiguration\_\_c cp1 =new Apttus\_Config2\_\_ProductConfiguration\_\_c(Name='test3',Apttus\_QPConfig\_\_Proposald\_\_c =testProposal.id);    
        insert cp1;  
         
         
        AST1.QuoteId = testProposal.id;      
     
        AST1.getQueryFilter(A1.Id);  
        AST1.getAssetSearchScope();  
        Apttus\_Config2.CustomClass.ActionParams params = new Apttus\_Config2.CustomClass.ActionParams();      
        AST1.getFilterExpr(params);   
        AST1.finish();      
    }  
    @isTest public static void test3(){  
        List<String> sc = null;  
        AssetLineItemCallback AST1 = new AssetLineItemCallback();  
        AST1.assetSearchFilter = null;  
        AST1.assetSearchScope = sc;  
        AST1.priceListId = null;  
        AST1.QuoteId =null;     
       Apttus\_Config2.CustomClass.ActionParams params = new Apttus\_Config2.CustomClass.ActionParams();      
       AST1.getFilterExpr(params);   
       Map<String, Schema.RecordTypeInfo> mapRType = Schema.SObjectType.Apttus\_Proposal\_\_Proposal\_\_c.getRecordTypeInfosByName();  
       Schema.RecordTypeInfo pRT = mapRType.get('OpCo');    
       Account A1 = new Account (Name='Test1');  
       Insert A1;    
          
       Apttus\_Proposal\_\_Proposal\_\_c testProposal = new Apttus\_Proposal\_\_Proposal\_\_c(RecordTypeID = pRT.getRecordTypeID());  
          
       testProposal.Apttus\_Proposal\_\_Account\_\_c = A1.Id;   
       insert testProposal;   
       AST1.QuoteId =testProposal.Id;  
       Apttus\_Config2.CustomClass.ActionParams params1 = new Apttus\_Config2.CustomClass.ActionParams();      
       AST1.getFilterExpr(params1);  
          
    }  
    @isTest Public static void test4(){  
          
        List<String> sc = null;  
        AssetLineItemCallback AST1 = new AssetLineItemCallback();  
        AST1.assetSearchFilter = null;  
        AST1.assetSearchScope = sc;  
        AST1.priceListId = null;  
        AST1.QuoteId =null;  
        Apttus\_Config2.ProductConfiguration c12 = null;  
        AST1.start(c12,AST1.assetSearchFilter,AST1.assetSearchScope);  
    }  
       
}

global class CartValidationCallback implements Apttus\_Config2.CustomClass.IValidationCallback2 {  
      
    global Apttus\_Config2.CustomClass.ValidationResult validateCart(Apttus\_Config2.ProductConfiguration configuration) {  
        Apttus\_Config2.CustomClass.ValidationResult result = new Apttus\_Config2.CustomClass.ValidationResult(true);  
          
        Apttus\_Config2\_\_ProductConfiguration\_\_c prodConfig = configuration.getConfigSO();  
          
        if (prodConfig.TPS\_Product\_\_c == true) { // don't check TPS quotes  
            return result;  
        }  
          
        //Boolean flag = false;  
        String errorMessage;  
          
        List<Apttus\_Config2.LineItem> allLines = configuration.getLineItems();  
        if (allLines.size() == 0) {  
            return result;  
        }  
          
        list<Apttus\_Config2\_\_LineItem\_\_c> lineItemList = getLineItems(allLines);  
          
        for(Apttus\_Config2\_\_LineItem\_\_c lineItem : lineItemList) {  
            if(lineItem.Sale\_Type\_\_c == null) {   
               result.isSuccess = false;  
               String msg = 'Sale Type not selected for ';  
                if (lineItem.Apttus\_Config2\_\_LineType\_\_c == 'Product/Service') {  
                    msg += 'product ' + lineItem.Apttus\_Config2\_\_ProductId\_\_r.Name;  
                }  
                else {  
                    msg += 'option ' + lineItem.Apttus\_Config2\_\_OptionId\_\_r.Name;  
                }  
                result.Messages.add(new apexpages.Message(Apexpages.Severity.Error, msg));  
            }  
        }  
          
        return result;  
    }  
      
    global Apttus\_Config2.CustomClass.ValidationResult validateAssetItems(Apttus\_Config2.ProductConfiguration cart, List<Apttus\_Config2\_\_TempRenew\_\_c> assetItems) {  
        Apttus\_Config2.CustomClass.ValidationResult result;  
        return result;  
    }     
      
    global Apttus\_Config2.CustomClass.ValidationResult validateRampLineItems(Apttus\_Config2.ProductConfiguration cart, List<Apttus\_Config2.LineItem> rampLineItems) {  
        Apttus\_Config2.CustomClass.ValidationResult result;  
        return result;  
    }  
      
    private static List<Apttus\_Config2\_\_LineItem\_\_C> getLineItems(List<Apttus\_Config2.LineItem> allLines) {  
        List<Apttus\_Config2\_\_LineItem\_\_C> lineItems = new List<Apttus\_Config2\_\_LineItem\_\_C>();  
        for (Apttus\_Config2.LineItem lineItemMO : allLines) {  
            lineItems.add(lineItemMO.getLineItemSO());  
        }  
        return lineItems;  
    }  
}

  private static Boolean isFieldValueModified(String fieldName, SObject newRecord, SObject oldRecord) {  
        if(newRecord != null && oldRecord != null && newRecord.get(fieldName) != oldRecord.get(fieldName)) {  
            return TRUE;  
        } else {  
            return FALSE;  
        }  
    }

   public static void CreateRenewalOpportunity(Map<Id, Apttus\_Config2\_\_Order\_\_c> newOrderMap, Map<Id, Apttus\_Config2\_\_Order\_\_c> oldOrderMap) {

        for(Id orderId : newOrderMap.keySet()) {  
            Apttus\_Config2\_\_Order\_\_c newOrder = newOrderMap.get(orderId);  
            Apttus\_Config2\_\_Order\_\_c oldOrder = oldOrderMap.get(orderId);                          
            if(newOrder.Apttus\_Config2\_\_Status\_\_c== 'Completed' && isFieldValueModified('Apttus\_Config2\_\_Status\_\_c', newOrder, oldOrder) && !newOrder.Is\_Change\_Order\_\_c   )                    
            {   
               if (System.isFuture() || System.isBatch()) {  
                 AddRenewalOpportunity(orderId);  
               }  
               else{  
                       system.debug('future method');   
                       AsynCreateRenewalOpportunity(orderId);   
               }  
  
            }  
        }  
        isCreateRenewalOpportunityExecuted = TRUE;  
                   
                  
    }

 //  Make call to sync opportunity asynchronous  
    @future  
    public static void AsynCreateRenewalOpportunity(String quoteId){  
        try {  
               AddRenewalOpportunity(quoteId);  
               }  
   
          
        catch(Exception e){  
            system.debug('Error during creation: ' + e.getMessage());  
        }  
     
    }

Check user name in   (UserInfo.getUserName() != 'sopcosystem@nttsecurity.com.dev1'

Use named credential for authentication when calling remote system.

Named credential – specify the callout endpoints URL and the authentication settings

Check named credential and authentication method – oauth or password authentication.

Post to create new, Put to updated, Get to request.

Method is future with callout = true

Create Https, HttpRequest, HttpsResponse objects.

Request.setheader ==           request.setHeader('Content-Type', 'application/json;charset=UTF-8');   
Request.setEndpoint ===  request.setEndpoint('callout:SNOWMSSP' + '?sysparm\_fields=u\_client\_id');  
Request.SetMethod ==           request.setMethod('POST');

Create jsonstring

request.setBody(jsonString);

if (Test.isRunningTest()) {  
              response.setStatus('OK');  
              response.setStatusCode(201);  
              response.setBody('{"u\_client\_id":"12345"}');  
else  
 Response = https.send(request);

Parse response - getstatuscode is 200 indicates that request was received and understood, code 201 indicates that account has been created successfully.

JSONParser.getbody used to parse the response, Parser,nexttoken to read elements

if (response.getStatusCode() == 201) {  
              JSONParser parser = JSON.createParser(response.getBody());  
              while (parser.nextToken() != null) {  
                  if ((parser.getCurrentToken() == JSONToken.FIELD\_NAME) &&  
                      (parser.getText() == 'u\_client\_id')) {  
                      // Get the value.  
                      parser.nextToken();  
                      tclientid = parser.getText();  
                      updateAccount = true;  
                  }  
              }

public class SNOWAccount  
{  
//class for integration into Servicenow  
  
    public static boolean runOnce = false;  
      
    @future (callout=true)  
    public static void SNOWAccountInsert(  string tname, string tid, string tbillingstreet, string tbillingpostalcode,   )  
            {  
      HttpRequest request = new HttpRequest();  
      Http http = new Http();      
      HttpResponse response = new HttpResponse();

      String requestNumber = '';  
      String requestSYSID = '';

      if (tclientid == NULL) {  
          request.setHeader('Accept', 'application/json');   
              tbillingstreet = tbillingstreet.replace('\n', '\\n');  
       
          String jsonString = '{"name":"'+tname+'"';  
          jsonString = jsonString + ',"u\_master\_id":"'+tid+'"';  
          jsonString = jsonString + ',"u\_opco":"'+topco+'"';          jsonString = jsonString + '}';   
    
          request.setEndpoint('callout:SNOWMSSP' + '?sysparm\_fields=u\_client\_id');  
          request.setMethod('POST');  
          request.setHeader('Content-Type', 'application/json;charset=UTF-8');   
          request.setBody(jsonString);  
          String clientID;  
         
          //  
          //  Create Account in MSSP SNOW  
          //  
          if (Test.isRunningTest()) {  
              response.setStatus('OK');  
              response.setStatusCode(201);  
              response.setBody('{"u\_client\_id":"12345"}');  
          }  
          else {      
              response = http.send(request);  
          }  
          //                
          // If the request is successful, parse the JSON response to get the client id value  
          //  
          if (response.getStatusCode() == 201) {  
              JSONParser parser = JSON.createParser(response.getBody());  
              while (parser.nextToken() != null) {  
                  if ((parser.getCurrentToken() == JSONToken.FIELD\_NAME) &&  
                      (parser.getText() == 'u\_client\_id')) {  
                      // Get the value.  
                      parser.nextToken();  
                      tclientid = parser.getText();  
                      updateAccount = true;  
                  }  
              }  
          }  
      }      
      //  
      //  If client id was passed in or it was just created then add the request in SNOW  
      //  
      system.debug ('TclientID: '+ tclientid);  
      if (tclientid != NULL) {   
          //  
          // First, determine if there are any MSS products.  If no MSS products  
          // exist in the quote then we won't add the SNOW request.  
          // If MSS products are found then build CSV file from proposal line items.  
          //  
          String csv = '';  
          boolean mss\_product\_found = false;  
          String proposalFileName = '';  
          String orderFileName = '';  
          Apttus\_Proposal\_\_Proposal\_\_c[] proposals;  
          Apttus\_Config2\_\_Order\_\_c[] orders;  
            
          system.debug('In client ID');  
                  

            system.debug('MSS Product Found: '+ mss\_product\_found);           
            if (mss\_product\_found) {                
              //  
              //  Get the sys\_id value of the SNOW MSSP client  
              //      
              HttpRequest request2 = new HttpRequest();  
**request2.setEndpoint('callout:SNOWMSSP' + '?sysparm\_query=u\_client\_id=' + tclientid + '&sysparm\_fields=sys\_id');**     
              request2.setHeader('Accept', 'application/json');   
              request2.setMethod('GET');  
              request2.setHeader('Content-Type', 'application/json;charset=UTF-8');   
              String clientSYSID;  
                
              if (Test.isRunningTest()) {  
                  response.setStatus('OK');  
                  response.setStatusCode(200);  
                  response.setBody('{"sys\_id":"1234567890"}');  
              }  
              else {  
                  response = http.send(request2);  
              }  
                                          
              JSONParser parser2 = JSON.createParser(response.getBody());  
              while (parser2.nextToken() != null) {  
                  if ((parser2.getCurrentToken() == JSONToken.FIELD\_NAME) &&  
                      (parser2.getText() == 'sys\_id')) {  
                       parser2.nextToken();  
                          clientSYSID = parser2.getText();  
                  }  
              }  
              //  
              //  Get the auto notifier contact from the SNOW MSSP account  
              //  
              request2.setEndpoint('callout:SNOWMSSPCONTACT' + '?sysparm\_query=account='+ clientSYSID +  
                                   '&first\_name=AutoNotifier' +  
                                   '&sysparm\_fields=sys\_id');     
              request2.setHeader('Accept', 'application/json');   
              request2.setMethod('GET');  
              request2.setHeader('Content-Type', 'application/json;charset=UTF-8');   
              String contactSYSID;  
                
              if (Test.isRunningTest()) {  
                  response.setStatus('OK');  
                  response.setStatusCode(200);  
                  response.setBody('{"sys\_id":"1234567890"}');  
              }  
              else {  
                  response = http.send(request2);  
              }     
                            
//            System.debug('user response body: ' + response.getBody());  
                            
                            
              JSONParser parser3 = JSON.createParser(response.getBody());  
              while (parser3.nextToken() != null) {  
                  if ((parser3.getCurrentToken() == JSONToken.FIELD\_NAME) &&  
                      (parser3.getText() == 'sys\_id')) {  
                       parser3.nextToken();  
                       contactSYSID = parser3.getText();  
                  }  
              }  
              //  
              //  Create request ticket in SNOW MSSP  
              //               
              String jsonString = '{"company":"'+clientSYSID+'"';  
              jsonString = jsonString + ',"requested\_for":"'+contactSYSID+'"';  
              jsonString = jsonString + ',"u\_request\_type":"24"';  
              if (ttype == 'New Business' || ttype == 'New') {  
                  //  On Board - New Client  
                  jsonString = jsonString + ',"u\_subcategory":"3acef8dd4f167380a41608111310c786"';  
              }  
              else  
              if (ttype == 'Renewal') {  
                  //  Renewals  
                  jsonString = jsonString + ',"u\_subcategory":"4e3fb4dd4f167380a41608111310c795"';  
              }  
              else {  
                  //  Change of Scope - Existing Client  
                  jsonString = jsonString + ',"u\_subcategory":"292ff0114f567380a41608111310c761"';  
              }  
              jsonString = jsonString + ',"u\_lgc\_internal":"true"';  
              jsonString = jsonString + ',"short\_description":"New order '+tordernumber+' received – Please process"';  
              jsonString = jsonString + ',"description":"Please process the new order attached"';  
              jsonString = jsonString + ',"impact":"3"';  
              jsonString = jsonString + ',"urgency":"3"';  
              jsonString = jsonString + ',"request\_state":"open"';  
//            jsonString = jsonString + ',"u\_waiting\_reason":"Waiting for OpCo"';  
              jsonString = jsonString + ',"assignment\_group":"a6b00bf74f7f134067e64a111310c7e5"';  
              jsonString = jsonString + '}';       
              system.debug('JSON STRING: '+jsonString);            
              request2.setEndpoint('callout:SNOWMSSPREQUEST'+'?sysparm\_fields=sys\_id,number');  
              request2.setMethod('POST');  
              request2.setHeader('Content-Type', 'application/json;charset=UTF-8');   
              request2.setBody(jsonString);  
                
              if (Test.isRunningTest()) {  
                  response.setStatus('OK');  
                  response.setStatusCode(201);  
                  response.setBody('{"sys\_id":"1234567890"}');  
              }  
              else {  
                  response = http.send(request2);  
              }  
  
//            System.debug('response body: ' + response.getBody());     
                        
              JSONParser parser4 = JSON.createParser(response.getBody());  
              while (parser4.nextToken() != null) {  
                  if ((parser4.getCurrentToken() == JSONToken.FIELD\_NAME)){  
                      if (parser4.getText() == 'sys\_id') {  
                          parser4.nextToken();  
                          requestSYSID = parser4.getText();  
                      }  
                      if (parser4.getText() == 'number') {  
                          parser4.nextToken();  
                          requestNumber = parser4.getText();  
                      }  
                        
                  }  
              }  
   
  
              //  
              //  Add CSV file as an attachment to the newly created request ticket in SNOW MSSP.  
              //  
              request2.setHeader('Accept', 'application/json');   
              request2.setEndpoint('callout:SNOWMSSPATTACHMENT' + '?table\_name=sc\_request&table\_sys\_id=' +  
                                  requestSYSID + '&file\_name=PO\_' + proposalFileName + '.csv');  
              request2.setMethod('POST');  
              request2.setHeader('Content-Type', 'text/csv');   
              request2.setBody(csv);  
              request2.setHeader('Accept', 'application/json');  
                
              if (Test.isRunningTest()) {  
                  response.setStatus('OK');  
                  response.setStatusCode(201);  
                  response.setBody('{"sys\_id":"1234567890"}');  
              }  
              else {  
                  response = http.send(request2);  
              }    
           }                         
      }         
      //  
      //  If a new account was created in SNOW MSSP, update the SNOW client id field in the SFDC account record.  
      //  
      if (updateAccount) {  
          Account accountToUpdate;  
          try {  
              accountToUpdate =   
              [SELECT SNOW\_Client\_ID\_\_c FROM Account  
               WHERE ID =:tid];  
              accountToUpdate.SNOW\_Client\_ID\_\_c = tclientid;  
              update accountToUpdate;  
          } catch (DmlException e) {  
              System.debug('An unexpected error has occurred: ' + e.getMessage());  
          }                      
      }  
      system.debug('Request Number: '+ requestNumber);  
      if (requestNumber != '') {  
      //removing code to update proposal SNOW Request ID.  Only on Order now.  
/\*          Opportunity oppoToUpdate;  
          try {  
              oppoToUpdate =   
              [SELECT SNOW\_Request\_Number\_\_c FROM Opportunity  
               WHERE ID =:toppid];  
              oppoToUpdate.SNOW\_Request\_Number\_\_c = requestNumber;  
              oppoToUpdate.SNOW\_Request\_ID\_\_c = requestSYSID;  
              update oppoToUpdate;  
          } catch (DmlException e) {  
              System.debug('An unexpected error has occurred: ' + e.getMessage());  
          } \*/            
          Apttus\_Config2\_\_Order\_\_c orderToUpdate;  
          try {  
              orderToUpdate =   
              [SELECT SNOW\_Request\_Number\_\_c FROM Apttus\_Config2\_\_Order\_\_c  
               WHERE ID =:torderid];  
              orderToUpdate.SNOW\_Request\_Number\_\_c = requestNumber;  
              orderToUpdate.SNOW\_Request\_ID\_\_c = requestSYSID;  
              update orderToUpdate;  
          } catch (DmlException e) {  
              System.debug('An unexpected error has occurred: ' + e.getMessage());  
          }   
      }  
       
   }  
      @future (callout=true)  
      public static void SNOWAccountUpdate(  string tname,  
                                             string tid,  
                                             string tbillingstreet,  
                                             string tbillingpostalcode,  
                                             string tbillingcity,                                      
                                             string tbillingstate,  
                                             string tbillingcountry,  
                                             string tclientid  
                                                                       
                                 )  
            
    {  
      if (tclientid != NULL) {  
          HttpRequest request = new HttpRequest();  
          request.setEndpoint('callout:SNOWMSSP' + '?sysparm\_query=u\_client\_id=' + tclientid + '&sysparm\_fields=sys\_id');  
          request.setMethod('GET');  
          request.setHeader('Accept', 'application/json');   
          Http http = new Http();  
          HttpResponse response = new HttpResponse();  
          if (Test.isRunningTest()) {  
              response.setStatus('OK');  
              response.setStatusCode(200);  
              response.setBody('{"sys\_id":"12345"}');  
          }  
          else {      
              response = http.send(request);  
          }  
        
          String sysid;  
          if (response.getStatusCode() == 200) {  
              JSONParser parser = JSON.createParser(response.getBody());  
              while (parser.nextToken() != null) {  
              if ((parser.getCurrentToken() == JSONToken.FIELD\_NAME) &&  
                  (parser.getText() == 'sys\_id')) {  
                  // Get the value.  
                  parser.nextToken();  
                  sysid = parser.getText();  
              }  
          }  
            
          if (response.getStatusCode() != 200 ||   
              sysid == NULL)   
              {  
              System.debug('Error retrieving SNOW MSSP customer ' + tclientid);  
              }  
          else {       
              if (tbillingstreet != NULL) {  
                  tbillingstreet = tbillingstreet.replace('\n', '\\n');  
                  tbillingstreet = tbillingstreet.replace('\r', '\\r');  
                  tbillingstreet = tbillingstreet.replace('\r\n', '\\r\\n');  
              }  
                
              String jsonString = '{"name":"'+tname+'"';  
              jsonString = jsonString + ',"street":"'+tbillingstreet+'"';  
              jsonString = jsonString + ',"city":"'+tbillingcity+'"';  
              jsonString = jsonString + ',"state":"'+tbillingstate+'"';  
              jsonString = jsonString + ',"zip":"'+tbillingpostalcode+'"';  
              jsonString = jsonString + ',"country":"'+tbillingcountry+'"';  
              jsonString = jsonString + ',"u\_master\_id":"'+tid+'"';  
              jsonString = jsonString + ',"u\_master\_system":"Salesforce"';  
              jsonString = jsonString + '}';   
    
              request.setEndpoint('callout:SNOWMSSP' + '/' + sysid);  
              request.setMethod('PUT');  
              request.setHeader('Content-Type', 'application/json;charset=UTF-8');   
              request.setBody(jsonString);  
   
              if (Test.isRunningTest()) {  
                  response.setStatus('OK');  
                  response.setStatusCode(200);  
                  response.setBody('{"sys\_id":"12345"}');  
              }  
              else {      
                  response = http.send(request);  
              }  
   
              // If the request is successful, parse the JSON response.  
              if (response.getStatusCode() != 200) {  
                  System.debug('An unexpected error has occurred when updating SNOW MSSP Client ' + tclientid + ' response code: ' + response.getStatusCode());  
              }  
          }  
      }  
   }  
}  
  
  
     @future (callout=true)  
      public static void SNOWRequestCancel( string toppid,  
                                            string tsysid,  
                                            string tsysnum,  
                                            string tordernumber )  
            
    {  
      if (tsysid != NULL && toppid != null) {  
          HttpRequest request = new HttpRequest();  
          request.setEndpoint('callout:SNOWMSSPREQUEST' + '/' + tsysid);  
          request.setMethod('PUT');  
          request.setHeader('Accept', 'application/json');   
          Http http = new Http();  
          HttpResponse response = new HttpResponse();  
                 
          String jsonString = '{"request\_state":"closed\_cancelled"';  
          jsonString = jsonString + ',"u\_cause\_notes":"SFDC Order '+tordernumber+' is cancelled"';  
          jsonString = jsonString + ',"u\_resolution\_code":"Cancelled"';  
          jsonString = jsonString + '}';   
  
          request.setHeader('Content-Type', 'application/json;charset=UTF-8');   
          request.setBody(jsonString);  
  
          if (Test.isRunningTest()) {  
              response.setStatus('OK');  
              response.setStatusCode(200);  
              response.setBody('{"sys\_id":"12345"}');  
          }  
          else {      
              response = http.send(request);  
          }  
  
          // If the request is successful, parse the JSON response.  
          if (response.getStatusCode() != 200) {  
              System.debug('An unexpected error has occurred when updating SNOW Request ' + tsysid + ' response code: ' + response.getStatusCode());  
          }  
            
      }  
   }  
}

Lightning web component to update status of order

updaterecord.js-meta.xml

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata" fqn="helloWorld">

    <apiVersion>45.0</apiVersion>

    <isExposed>true</isExposed>

    <targets>

        <target>lightning\_\_RecordPage</target>

    </targets>

</LightningComponentBundle>

Html

<template>

<lightning-card title="Update Account Record" icon-name="standard:record">

<div class="slds-m-around\_medium">

<lightning-input label="Name" onchange={handleNameChange} class="slds-m-bottom\_x-small"></lightning-input>

<lightning-input label="Annual Revenue" type="number" onchange={handleNameChange} class="slds-m-bottom\_x-small"></lightning-input>

<lightning-button label="Update Account" variant="brand" onclick={updateAccount}></lightning-button>

</div>

</lightning-card>

</template>

Create Javascript

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

import {ShowToastEvent} from 'lightning/platformShowToastEvent';

import ACCOUNT\_OBJECT from '@salesforce/schema/Account';

import NAME\_FIELD from '@salesforce/schema/Account.Name';

import ID\_FIELD from '@salesforce/schema/Account.ID';

import ANNUALREV\_FIELD from '@salesforce/schema/Account.AnnualRevenue';

import Fax\_FIELD from '@salesforce/schema/Account.Fax';

import PHONE\_FIELD from '@salesforce/schema/Account.Phone';

import {getRecord,updateRecord,generateRecordInputForUpdate,getFieldValue,} from 'lightning/uiRecordApi';

import {CurrentPageReference} from 'lightning/navigation';

export default class Updaterecord extends LightningElement {

@api recordId;

@track accountId;

@track name = 'Demo Account';

@track annualrev = 0;

@track fax = '1231234345345';

@track phone = '1231234345345';

handleNameChange(event) {

this.accountId = undefined;

console.log('label values --->>' + event.target.label);

if (event.target.label === 'Name') {

this.name = event.target.value;

}

if (event.target.label === 'Annual Revenue') {

this.annualrev = event.target.value;

}

if (event.target.label === 'Fax') {

this.fax = event.target.value;

}

if (event.target.label === 'Phone') {

this.phone = event.target.value;

}

}

updateAccount() {

let record = {

fields: {

Id: this.recordId,

Name: this.name,

AnnualRevenue:this.annualrev ,

Fax:this.fax ,

Phone:this.phone,

},

};

updateRecord(record)

// eslint-disable-next-line no-unused-vars

.then(() => {

this.dispatchEvent(

new ShowToastEvent({

title: 'Success',

message: 'Record Is Updated',

variant: 'sucess',

}),

);

})

.catch(error => {

this.dispatchEvent(

new ShowToastEvent({

title: 'Error on data save',

message: error.message.body,

variant: 'error',

}),

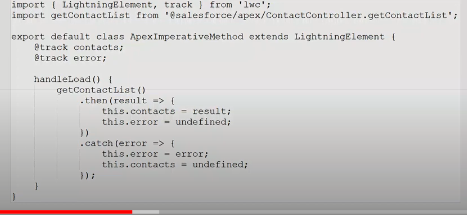
);

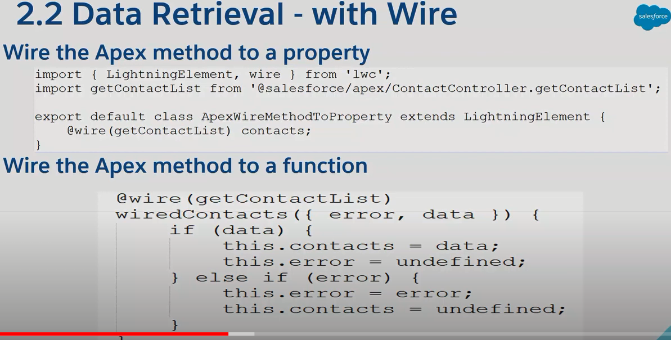
});

}

}

<https://www.youtube.com/watch?v=DfV7WnyEPTE>





Lightning web components can import methods from Apex classes. The imported methods are functions that the component can call either via @wire or imperatively.

See whether a base Lightning component, like lightning-record-form, lightning-record-view-form, or lightning-record-edit-form works for your use case. If they don’t give you enough flexibility, use a wire adapter like [getListUi](https://developer.salesforce.com/docs/component-library/documentation/en/lwc/reference_get_list_ui.html" \o "Use this wire adapter to get the records and metadata for a list view.) or [getRecordUi](https://developer.salesforce.com/docs/component-library/documentation/en/lwc/reference_wire_adapters_record_ui.html" \o "Use this wire adapter to get layout information, metadata, and data to build UI for one or more records.). And if you can’t use a wire adapter, write an Apex method. For example, to return a simple list of accounts, it would be best to use getListUi. But to use a SOQL query to select certain accounts, you have to use an Apex method.

To call an Apex method, a Lightning web component can:

* Wire a property
* Wire a function
* Call a method imperatively

To expose an Apex method to a Lightning web component, the method must be [static](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_static.htm) and either [global or public](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_access_modifiers.htm). Annotate the method with @AuraEnabled.(Catchabe=true)

public with sharing class ContactController {

@AuraEnabled(cacheable=true)

public static List<Contact> getContactList() {

return [

SELECT Id, Name, Title, Phone, Email, Picture\_\_c

FROM Contact

WHERE Picture\_\_c != null

WITH SECURITY\_ENFORCED

LIMIT 10

];

}

}

To call an Apex method imperatively, you can choose to set cacheable=true. This setting caches the result on the client, and prevents Data Manipulation Language (DML) operations.

Use the Continuation class in Apex to make a long-running request to an external Web service.

As we’ve learned, to return a simple list of contacts, it’s best to use [getListUi](https://developer.salesforce.com/docs/component-library/documentation/en/lwc/reference_get_list_ui.html" \o "Use this wire adapter to get the records and metadata for a list view.)

import { LightningElement, wire } from 'lwc';

import getContactList from '@salesforce/apex/ContactController.getContactList';

export default class ApexWireMethodToProperty extends LightningElement {

@wire(getContactList) contacts;

}

Now let’s wire an Apex method that takes a parameter. This component is also in the [lwc-recipes](https://github.com/trailheadapps/lwc-recipes" \t "_blank) repo

// ContactController.cls

public with sharing class ContactController {

@AuraEnabled(cacheable=true)

public static List<Contact> findContacts(String searchKey) {

String key = '%' + searchKey + '%';

return [

SELECT Id, Name, Title, Phone, Email, Picture\_\_c

FROM Contact

WHERE Name LIKE :key AND Picture\_\_c != null

WITH SECURITY\_ENFORCED

LIMIT 10

];

}

}

*// apexWireMethodWithParams.js*

import { LightningElement, wire } from 'lwc';

import findContacts from '@salesforce/apex/ContactController.findContacts';

*/\*\* The delay used when debouncing event handlers before invoking Apex. \*/*

const DELAY = 300;

export default class ApexWireMethodWithParams extends LightningElement {

searchKey = '';

**@wire(findContacts, { searchKey: '$searchKey' })**

contacts;

handleKeyChange(event) {

*// Debouncing this method: Do not update the reactive property as long as this function is*

*// being called within a delay of DELAY. This is to avoid a very large number of Apex method calls.*

window.clearTimeout(this.delayTimeout);

const searchKey = event.target.value;

this.delayTimeout = setTimeout(() => {

this.searchKey = searchKey;

}, DELAY);

}

}

The template uses searchKey as the value of the <lightning-input> field.

*<!-- apexWireMethodWithParams.html -->*

<template>

<lightning-card title="ApexWireMethodWithParams" icon-name="custom:custom63">

<div class="slds-m-around\_medium">

<lightning-input type="search" onchange={handleKeyChange} class="slds-m-bottom\_small" label="Search" **value={searchKey}**></lightning-input>

<template if:true={contacts.data}>

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

<p key={contact.Id}>{contact.Name}</p>

</template>

</template>

<template if:true={contacts.error}>

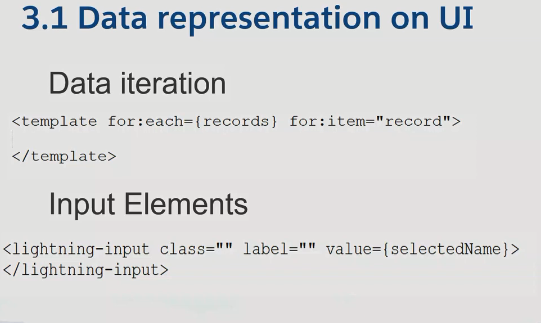
<c-error-panel errors={contacts.error}></c-error-panel>

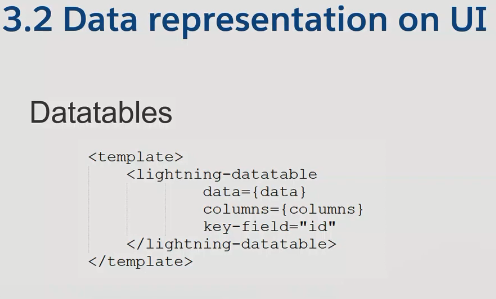
</template>

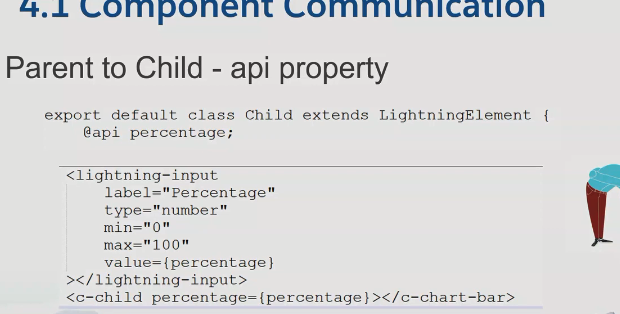
</div>

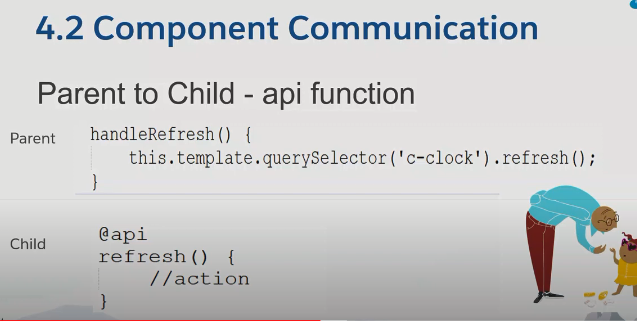
</lightning-card>

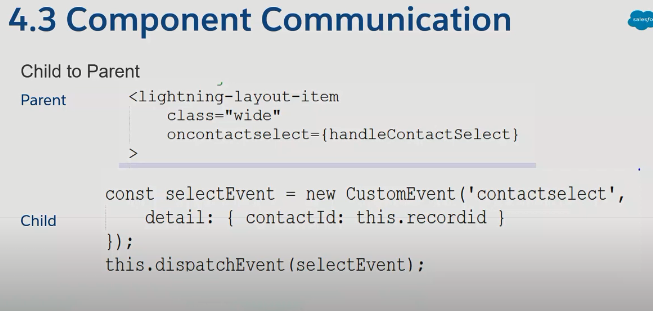
</template>

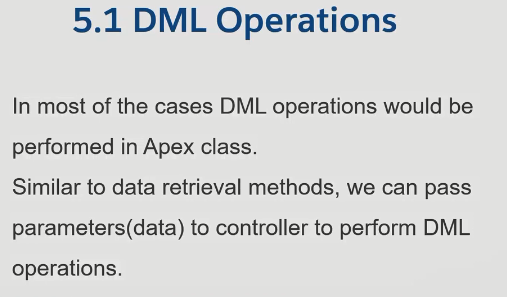


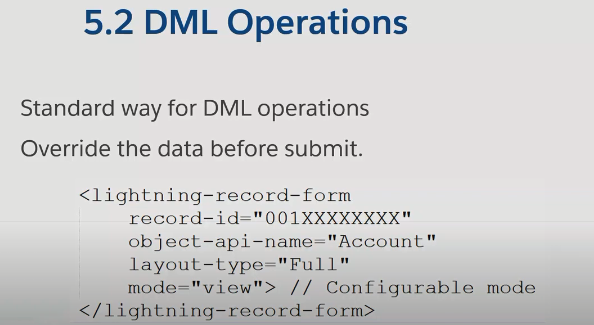


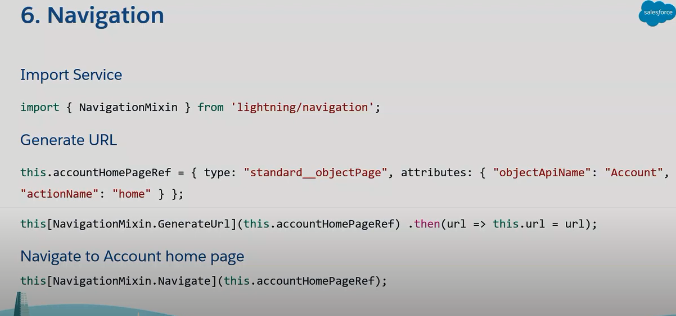


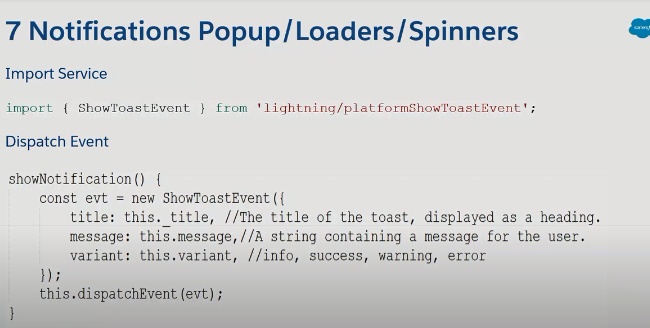


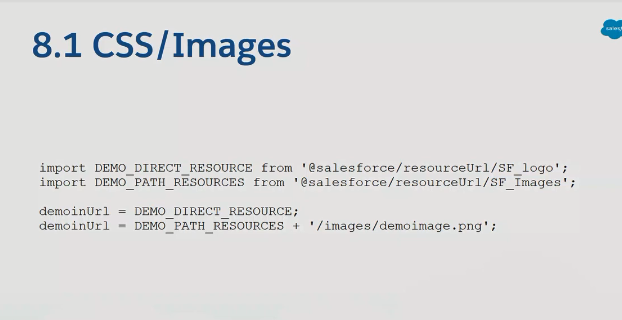


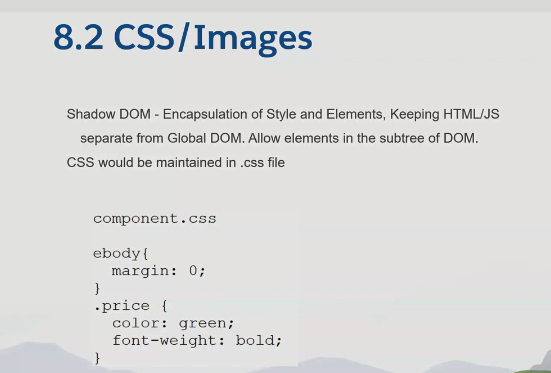


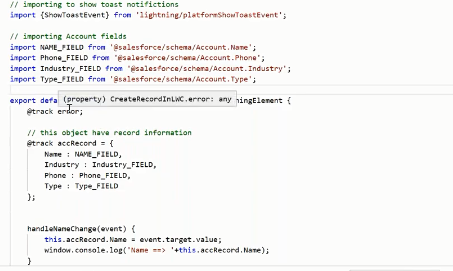


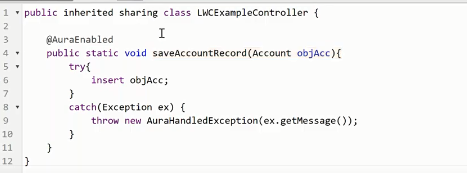














<https://www.youtube.com/watch?v=1NtWFCAUB0c>