DocFynd Injection Service – Document List Service

About

This document explains how to configure a custom document list.

Introduction

- DocFynd injection service serves the custom need of businesses. It provides an easy way for salesforce developers to change the behavior of the DocFynd application by adding custom logic via apex code. They can ask the developer/salesforce admin to write a custom code that will enable custom behavior.
- DocFynd uses a dependency injection service to enable this customization.

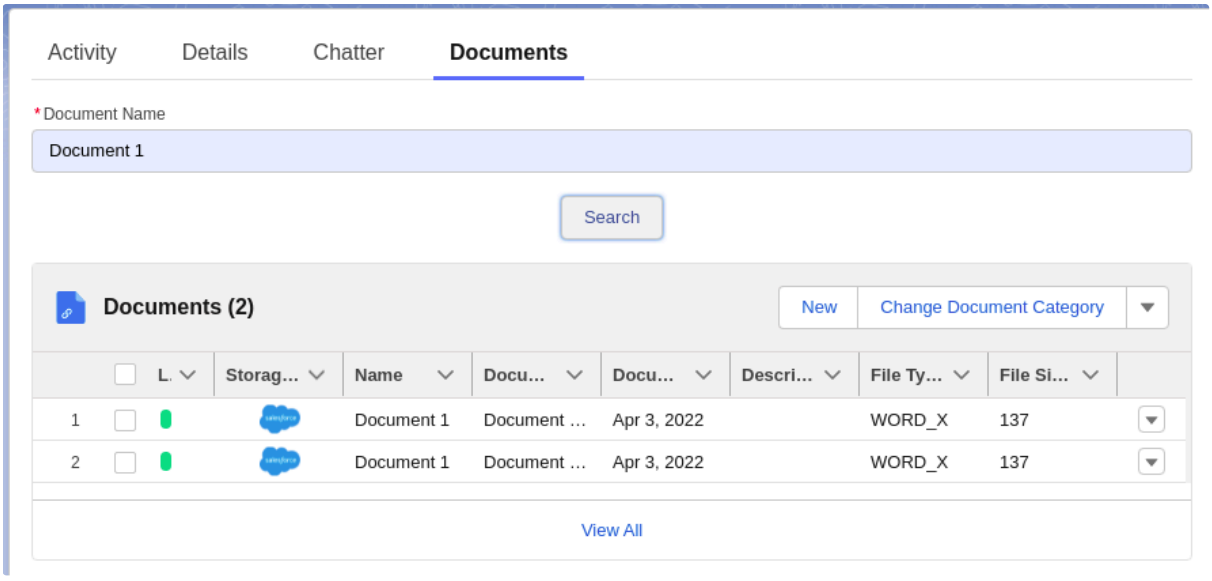
How does it work!

- We have used the salesforce callable interface which looks for some specific class names while executing the default functionality of our package.
- We have added callable interfaces to display a list of documents in the document list component. For e.g., You can customize what documents should be listed in the document list table according to your requirements.

Sample use cases and implementations

Filter Documents List using Aura component

When a document name is entered and submitted, documents listed in the document list table are filtered according to the values entered.



Steps to achieve this use case

1. Create an aura component using the code below.

```
<aura:component implements="force:appHostable,flexipage:availableForAllPageTypes,flexipage:availableForRecord
    <aura:attribute name="recordId" type="Id" />
    <aura:attribute name="noOfRecToDisplay" type="Integer" default="5" />
    <aura:attribute name="displayActions" type="Boolean" default="true" />
```

```
    <aura:attribute name="sObjectName" type="String"  />
    <lightning:input style="padding-bottom : 2%;" aura:id="Name" name="DocumentName" label="Document Name" re
    <div style="padding-bottom : 2%;">
    <lightning:button class="slds-align_absolute-center" label="Search" onclick="{! c.callLwc }"/>
    </div>

    <docfynd:documentList aura:id="documentList" recordId="{!v.recordId}" objectApiName="{!v.sObjectName}"
                        noOfRecToDisplay="{!v.noOfRecToDisplay}" displayActions="{!v.displayActions}"/>
</aura:component>
```

2. And enter the below code in the aura component controller.

```
({
    callLwc : function(component, event, helper) {
        var name = component.find("Name").get("v.value");
        var paramList =
            {"Name":name}
        ;
        component.set("v.parameterList", paramList);
        var findDocListComp=component.find('documentList');
        findDocListComp.refreshDocTable(paramList);

    }
})
```

3. Pass the parameters to the document list component as given in the code.
4. Create a callable apex method using the code below and get your parameters as arg and query based on the entered parameters.

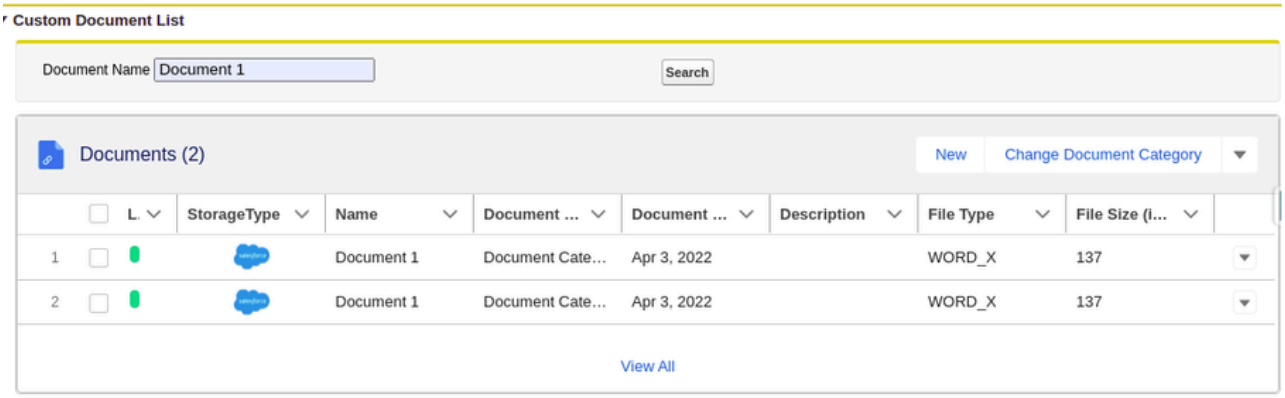
```
global class DocFyndDocListService implements callable {
    global object call(String action,Map<String , Object>args){
        docfynd.InjectionServiceRes injectionServiceRes = new docfynd.InjectionServiceRes();
        switch on action{
            when 'customDocList'{
                System.debug('args==> '+args); // Get parameter list as args that you are passing to the comp
                // You can write your custom logic based on the parameter list
                Integer offset = (Integer.valueOf(args.get('pageNumber')) - 1) * Integer.valueOf(args.get('no

                List<docfynd__Document__c> docList =[SELECT Id, docfynd__Label__c, Name, docfynd__DocumentCat
                                                    docfynd__FileType__c, docfynd__FileSize__c, docfynd__Docu
                                                    FROM docfynd__Document__c WHERE Name =: string.valueOf(ar

                List<docfynd__Document__c> documentList =[SELECT Id, docfynd__Label__c, Name, docfynd__Docume
                                                    docfynd__FileType__c, docfynd__FileSize__c, docfynd__Docu
                                                    FROM docfynd__Document__c WHERE Name =: string.valueOf(ar
                Map<String, Object> responseMap = new Map<String, Object>();
                responseMap.put('docList', docList);
                responseMap.put('totalRecords', documentList.size());

                injectionServiceRes.response = responseMap;
                injectionServiceRes.isActionExists = true;
                return injectionServiceRes;
            }
        }
        return null;
    }
}
```

5. Make sure the class name is entered as the same as in the code snippet or else the manage package won’t find your callable apex class.

6. Make sure you have return the document list and total record count in the object parameters “docList" and "totalRecords" as in the code snippet.
7. Drag and drop the aura component that you have created in the record page in which you want the component to be viewed and click on save.
8. Now you will find your component on the record page.
9. Now enter the Document name and click on submit, then the documents with the entered name will be shown in the document list.

Filter Documents List using Visualforce page

When a document name is entered and searched, documents listed in the document list table are filtered according to the values entered.



Steps to achieve this use case

1. Create a Visualforce page using the code below.

```
<apex:page showHeader="true" sidebar="true" lightningStyleSheets="true" standardController="Opportunity" exte
    <apex:form >
        <apex:pageBlock >
            <apex:pageBlockSection >
                <label for="docName">Document Name</label>
                    <input title="Document Name" id="docName" value="" type="text"/>
                <apex:commandButton onclick="callDocumentList()" value="Search" onComplete="return null;"/>
            </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
    <apex:includeLightning />

    <div id="LightningComponentid" />

<!-- the Id of div tag which will be used to render your LWC component  -->
    <script>

            $Lightning.use("docfynd:docListApp", function() {
            $Lightning.createComponent("docfynd:documentList",
            {
                recordId : '{!$CurrentPage.parameters.id}',
                noOfRecToDisplay : 5,
                displayActions : true,
                isVFPage : true,
                objectApiName : '{!objApiName}'

            },
```

```
            "LightningComponentid", // the Id of div tag where your component will be rendered
            function(cmp) {});
        });
    function callDocumentList(event){
        var name = document.getElementById("docName").value;
        var paramList = {
            "Name": name
        };
        var documentList = document.querySelector("docfynd-document-list");
        documentList.refreshDocTable(paramList);
    }
    </script>
</apex:page>
```

2. Pass the parameters to the document list component.
3. Create an apex class using the code below.

```
public with sharing class DocUploadVFController {

    public Id recordId {get; set;}
    public String objApiName {get; set;}

    public DocUploadVFController(ApexPages.StandardController stdController){
        recordId = ApexPages.CurrentPage().getparameters().get('id');
        objApiName = recordId.getSObjectType().getDescribe().getName();
    }
}
```

4. Now create a VF page by using the code below.

```
<apex:page showHeader="true" sidebar="true" lightningStyleSheets="true" standardController="Opportunity" exte
    <apex:includeLightning />
    <div id="LightningComponentid" />
<!-- the Id of div tag which will be used to render your LWC component  -->
    <script>
        if('{!$CurrentPage.parameters.isRelatedList}'){
            let paramListDecoded = JSON.parse(window.atob('{!$CurrentPage.parameters.parameterList}'));
            $Lightning.use("docfynd:docListApp", function() {
            $Lightning.createComponent("docfynd:documentList",
            {
                recordId : '{!$CurrentPage.parameters.id}',
                noOfRecToDisplay : '{!$CurrentPage.parameters.noOfRecToDisplay}',
                isVFPage : true,
                displayActions : '{!$CurrentPage.parameters.displayActions}',
                isRelatedList : '{!$CurrentPage.parameters.isRelatedList}',
                isDocListComp : '{!$CurrentPage.parameters.isDocListComp}',
                parameterList : paramListDecoded,
                objectApiName : '{!objApiName}'

            },
            "LightningComponentid", // the Id of div tag where your component will be rendered
            function(cmp) {});
        });
        } else {
            $Lightning.use("docfynd:docListApp", function() {
            $Lightning.createComponent("docfynd:documentList",
            {
                recordId : '{!$CurrentPage.parameters.id}',
                noOfRecToDisplay : 5,
```

```
                isVFPage : true,
                displayActions : true,
                objectApiName : '{!objApiName}'
            },
            "LightningComponentid", // the Id of div tag where your component will be rendered
            function(cmp) {});
        });
        }
    </script>
</apex:page>
```

5. Create a callable apex method using the code below and get your parameters as arg and query based on the entered parameters.

```
global class DocFyndDocListService implements callable {
    global object call(String action,Map<String , Object>args){
        docfynd.InjectionServiceRes injectionServiceRes = new docfynd.InjectionServiceRes();
        switch on action{
            when 'customDocList'{
                System.debug('args==> '+args); // Get parameter list as args that you are passing to the comp
                // You can write your custom logic based on the parameter list
                Integer offset = (Integer.valueOf(args.get('pageNumber')) - 1) * Integer.valueOf(args.get('no

                List<docfynd__Document__c> docList =[SELECT Id, docfynd__Label__c, Name, docfynd__DocumentCat
                                                     docfynd__FileType__c, docfynd__FileSize__c, docfynd__Docu
                                                     FROM docfynd__Document__c WHERE Name =: string.valueOf(ar

                List<docfynd__Document__c> documentList =[SELECT Id, docfynd__Label__c, Name, docfynd__Docume
                                                     docfynd__FileType__c, docfynd__FileSize__c, docfynd__Docu
                                                     FROM docfynd__Document__c WHERE Name =: string.valueOf(ar
                Map<String, Object> responseMap = new Map<String, Object>();
                responseMap.put('docList', docList);
                responseMap.put('totalRecords', documentList.size());

                injectionServiceRes.response = responseMap;
                injectionServiceRes.isActionExists = true;
                return injectionServiceRes;
            }
        }
        return null;
    }
}
```

6. Make sure the class name is entered as the same as in the code snippet or else the manage package won't find your callable apex class.
7. Make sure you have return the document list and total record count in the object parameters “docList" and "totalRecords" as in the code snippet.
8. Drag and drop the Visualforce page that you have created in the page layout and click on save.
9. Now you will find your Visualforce page on your record page.
10. Enter any document name and click on search, then the documents with the entered name will be shown in the document list.