GUIDE

Nov. 2020

How to Create a Translation Framework for a Custom Salesforce App

oktana

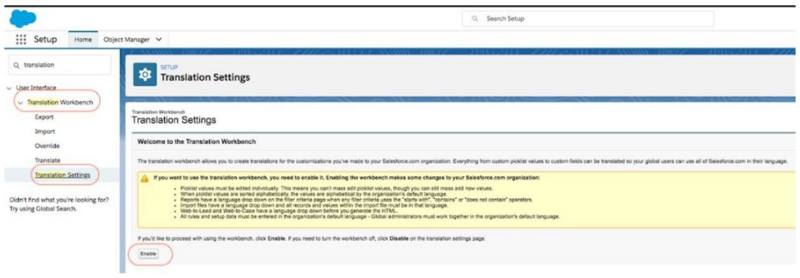
INDEX

With global growth, localization is more important than ever. For Salesforce AppExchange apps, it is possible to automatically translate anything that is standard. However, custom development needs to be translated manually.

One of our Salesforce experts, who has been working with Salesforce technologies for 12 years, has put together a step-by-step guide on how to create a translation framework for custom Salesforce AppExchange apps.

How to Enable Translations in Salesforce

The first thing to take into consideration is enabling translations from **Setup** → **Translation Workbench** → **Translation Language Settings**:



Translation Workbench
Translation Settings

Welcome to the Translation Workbench

The translation workbench allows you to create translations for the customizations you've made to your Salesforce.com organization. Everything from custom picklist values to custom fields can be translated so your global users can use all of Salesforce.com in their language.

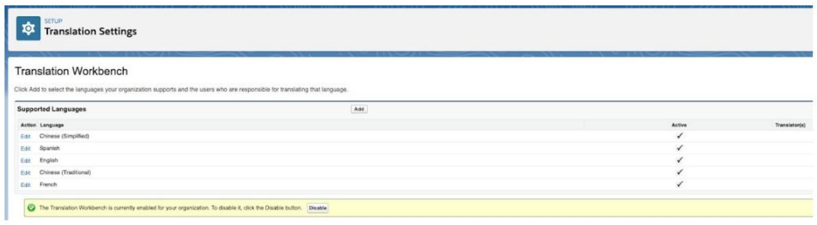
If you want to use the translation workbench, you need to enable it. Enabling the workbench makes some changes to your Salesforce.com organization:

- Picklist values must be edited individually. This means you can't mass edit picklist values, though you can still mass add new values.
- When picklist values are sorted alphabetically, the values are alphabetical by the organization's default language.
- Reports have a language drop down on the filter criteria page when any filter criteria uses the "starts with", "contains" or "does not contain" operators.
- Import files have a language drop down and all records and values within the import file must be in that language.
- Web-to-Lead and Web-to-Case have a language drop down before you generate the HTML.
- All rules and setup data must be entered in the organization's default language - Global administrators must work together in the organization's default language.

If you'd like to proceed with using the workbench, click **Enable**. If you need to turn the workbench off, click **Disable** on the translation settings page.

Enable

Click on the **Enable** button to enable the **Translation Workbench**



Translation Workbench

Click Add to select the languages your organization supports and the users who are responsible for translating that language.

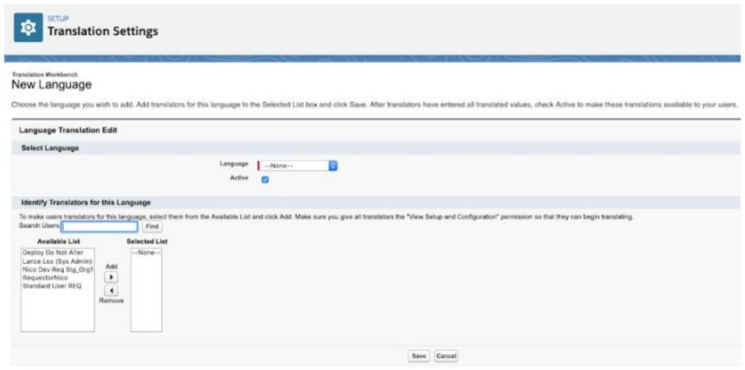
Supported Languages Add

Action	Language	Active	Translator(s)
Edit	Chinese (Simplified)	✓	
Edit	Spanish	✓	
Edit	English	✓	
Edit	Chinese (Traditional)	✓	
Edit	French	✓	

The Translation Workbench is currently enabled for your organization. To disable it, click the Disable button. Disable

Languages added to **Translation Workbench**

Once the Translation Workbench is enabled, you have the ability to add the languages you'd like available for translation. Click the **Add** button and select the language from the **Language** picklist.



Translation Workbench
New Language

Choose the language you wish to add. Add translators for this language to the Selected List box and click Save. After translators have entered all translated values, check Active to make these translations available to your users.

Language Translation Edit

Select Language

Language: --None--
Active: ✓

Identify Translators for this Language

To make users translators for this language, select them from the Available List and click Add. Make sure you give all translators the "View Setup and Configuration" permission so that they can begin translating.

Search Users: Find

Available List		Selected List
Deploy Do Not Alter Lance Los (Sys Admin) Nico Dev Req Stg_Org1 RequestorNico Standard User REQ	Add Remove	--None--

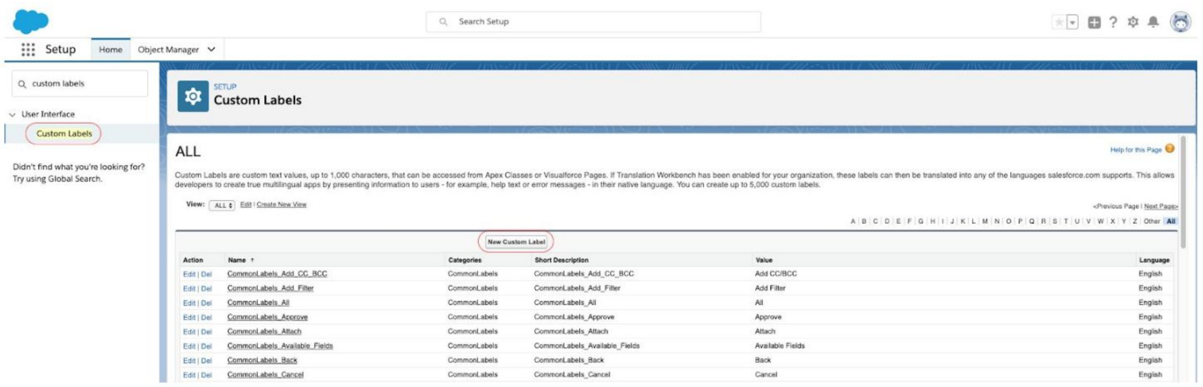
Save Cancel

Leave the **Active** checkbox selected and leave the last section as is. Click the **Save** button.

NOTE: If you do not enable the Translation Workbench in the target org and deploy translation-related metadata, it may trigger the following error:
translations/es.translation -- Error: Not available for deploy for this organization

How to Create Custom Labels in Salesforce

In order to create a **custom label**, go to **Setup**→ in **Quick Find**, search for **"*custom labels*"**:



Custom Labels are custom text values, up to 1,000 characters, that can be accessed from Apex Classes or Visualforce Pages. If Translation Workbench has been enabled for your organization, these labels can then be translated into any of the languages salesforce.com supports. This allows developers to create true multilingual apps by presenting information to users - for example, help text or error messages - in their native language. You can create up to 5,000 custom labels.

View: ALL | Edit | Create New View

<Previous Page | Next Page>

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

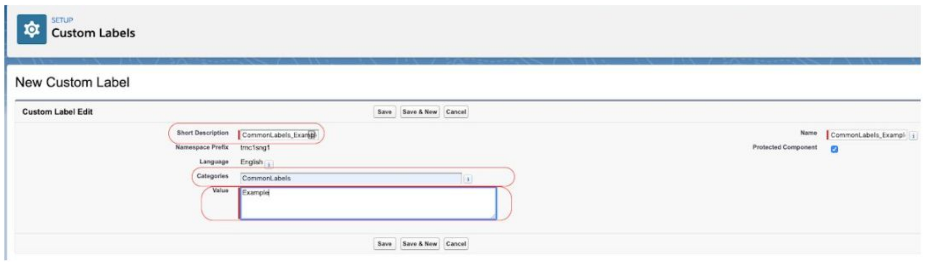
New Custom Label

Action	Name ↑	Categories	Short Description	Value	Language
Edit | Del	CommonLabels_Add_CC_BCC	CommonLabels	CommonLabels_Add_CC_BCC	Add CC/BCC	English
Edit | Del	CommonLabels_Add_Filter	CommonLabels	CommonLabels_Add_Filter	Add Filter	English
Edit | Del	CommonLabels_All	CommonLabels	CommonLabels_All	All	English
Edit | Del	CommonLabels_Approve	CommonLabels	CommonLabels_Approve	Approve	English
Edit | Del	CommonLabels_Attach	CommonLabels	CommonLabels_Attach	Attach	English
Edit | Del	CommonLabels_Available_Fields	CommonLabels	CommonLabels_Available_Fields	Available Fields	English
Edit | Del	CommonLabels_Back	CommonLabels	CommonLabels_Back	Back	English
Edit | Del	CommonLabels_Cancel	CommonLabels	CommonLabels_Cancel	Cancel	English

Press **New Custom Label** button to create a custom label

Considerations when creating a custom label:

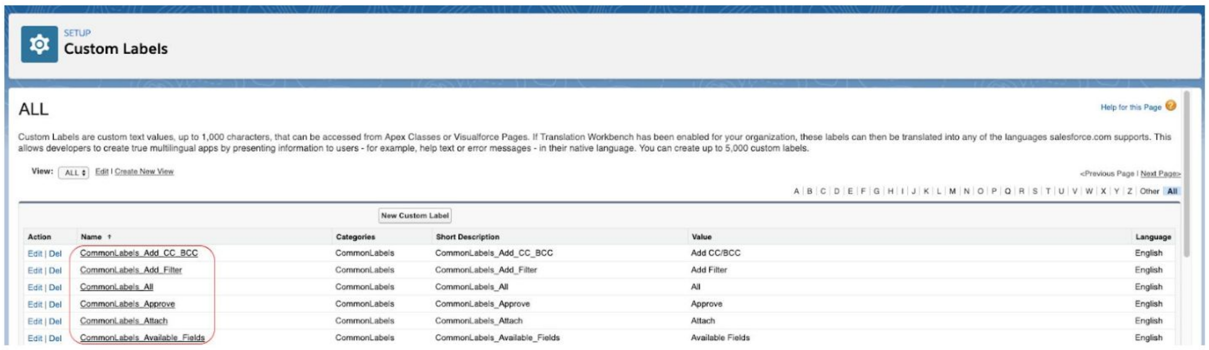
1. Before creating a custom label, please search the current labels to confirm there are no existing labels that meet your needs.
2. Use your common sense and knowledge about the app to determine if the label you are about to create might be used multiple times or if it is too specific (for example, a "Send" button vs. "Approve and Retest" button). Ideally, choose a label that can be reused.
3. Next, there is a naming convention established which is as follows **<Placement>Labels_<Name_Of_The_Label_Separated_With_Underscore>**. Where **"Placement"** is the object to which the label belongs.
   - The name of the placement must start with a capital letter and each word must be concatenated, also starting with a capital letter.
   - At the end it must include **"*Labels*"** as in the following example: if we are creating a label for a Request page and we know it most likely isn't used anywhere else, the prefix would be **CreateRequestLabels**. On the other hand, if it will be reused, the prefix must be **CommonLabels**.
   - The suffix would be a mnemotechnic name of the label to be translated. If we continue with the previous example and need to create a label **"*Request Name*"** on the Request page, the suffix would be **Request_Name**.
   - Altogether, prefix and suffix will end up like this: **CreateRequestLabels_Request_Name**
4. In the **Category** field you must enter the prefix mentioned in the previous point. This helps group the labels within Apex components. (ex. CreateRequestLabels)
5. In the **Value** field enter the translated text, which the label will translate to at the Visualforce page level.



Short Description	CommonLabels_Exampl
Namespace Prefix	tmc1sng1
Language	English
Categories	CommonLabels
Value	Example
Name	CommonLabels_Exampl
Protected Component	✓

How to Translate Labels in Salesforce

In order to translate a label, we need to go to the main table where all custom labels are displayed **Setup** → and in **Quick Find** search for **"*custom labels*"**. Select the desired label by clicking on its name.

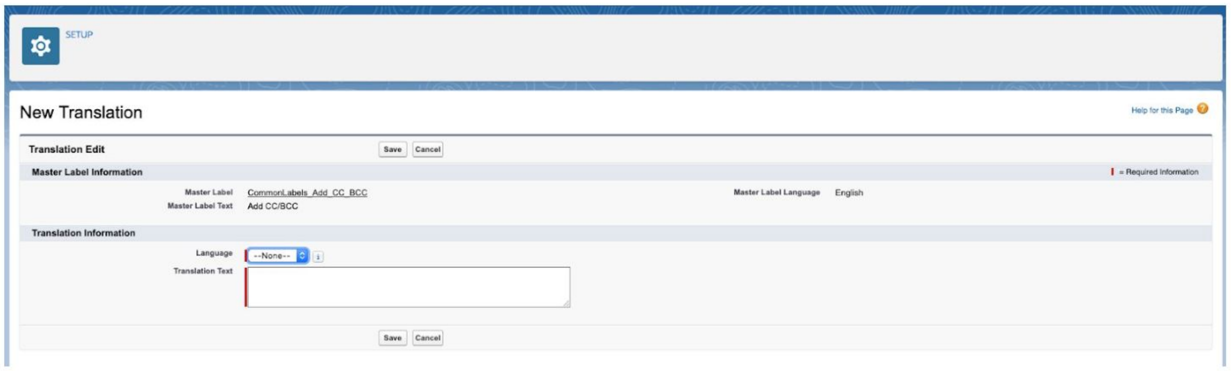

SETUP
Custom Labels
Help for this Page
View: ALL
Edit | Create New View
<Previous Page | Next Page>
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All
New Custom Label

ALL

Custom Labels are custom text values, up to 1,000 characters, that can be accessed from Apex Classes or Visualforce Pages. If Translation Workbench has been enabled for your organization, these labels can then be translated into any of the languages salesforce.com supports. This allows developers to create true multilingual apps by presenting information to users - for example, help text or error messages - in their native language. You can create up to 5,000 custom labels.

Action	Name ↑	Categories	Short Description	Value	Language
Edit | Del	CommonLabels_Add_CC_BCC	CommonLabels	CommonLabels_Add_CC_BCC	Add CC/BCC	English
Edit | Del	CommonLabels_Add_Filter	CommonLabels	CommonLabels_Add_Filter	Add Filter	English
Edit | Del	CommonLabels_All	CommonLabels	CommonLabels_All	All	English
Edit | Del	CommonLabels_Approve	CommonLabels	CommonLabels_Approve	Approve	English
Edit | Del	CommonLabels_Attach	CommonLabels	CommonLabels_Attach	Attach	English
Edit | Del	CommonLabels_Available_Fields	CommonLabels	CommonLabels_Available_Fields	Available Fields	English

On the **New Translation** window, we can add a translation for the selected label. Select the language you are translating from and enter the translated text (this is how the **Master Label Text** label translates to the selected language).

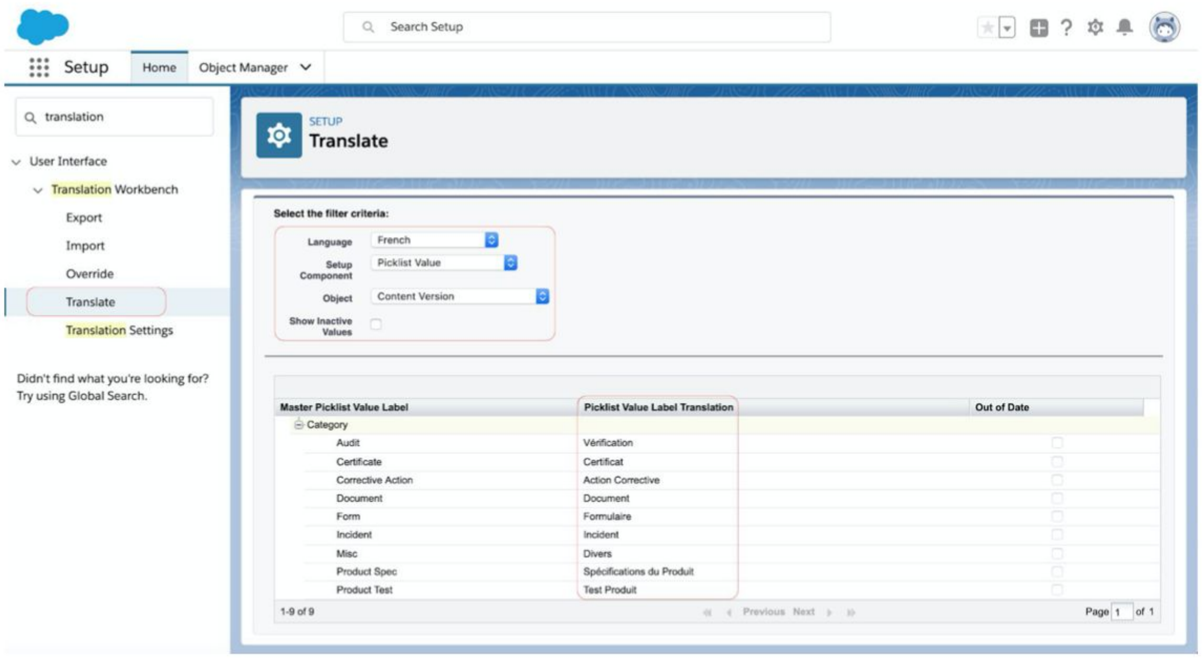


Master Label	CommonLabels_Add_CC_BCC
Master Label Text	Add CC/BCC
Master Label Language	English
Language	--None--
Translation Text	

NOTE: It is important to take into account that **Master Label Language** should always be set to **"*English*"**. This avoids any possible issues that might occur if the User or Admin changes the language to one that isn't added to the Translation Workbench and of course has not yet been translated. What happens then? If the language selected isn't in the Translation Workbench, the org will display the custom labels in English because it can't find any translations for the untranslated language.

How to Translate a Picklist in Salesforce

In order to translate a picklist, we need to go to, **Setup** → in **Quick Find**, search for **"*translate*"**.



Select the filter criteria:

Language: French
Setup Component: Picklist Value
Object: Content Version
Show Inactive Values: ☐

Master Picklist Value Label	Picklist Value Label Translation	Out of Date
Category		
Audit	Vérification	☐
Certificate	Certificat	☐
Corrective Action	Action Corrective	☐
Document	Document	☐
Form	Formulaire	☐
Incident	Incident	☐
Misc	Divers	☐
Product Spec	Spécifications du Produit	☐
Product Test	Test Produit	☐

1-9 of 9 Previous Next Page 1 of 1

The upper section offers several fields which need to be described in order to carry out the required translations:

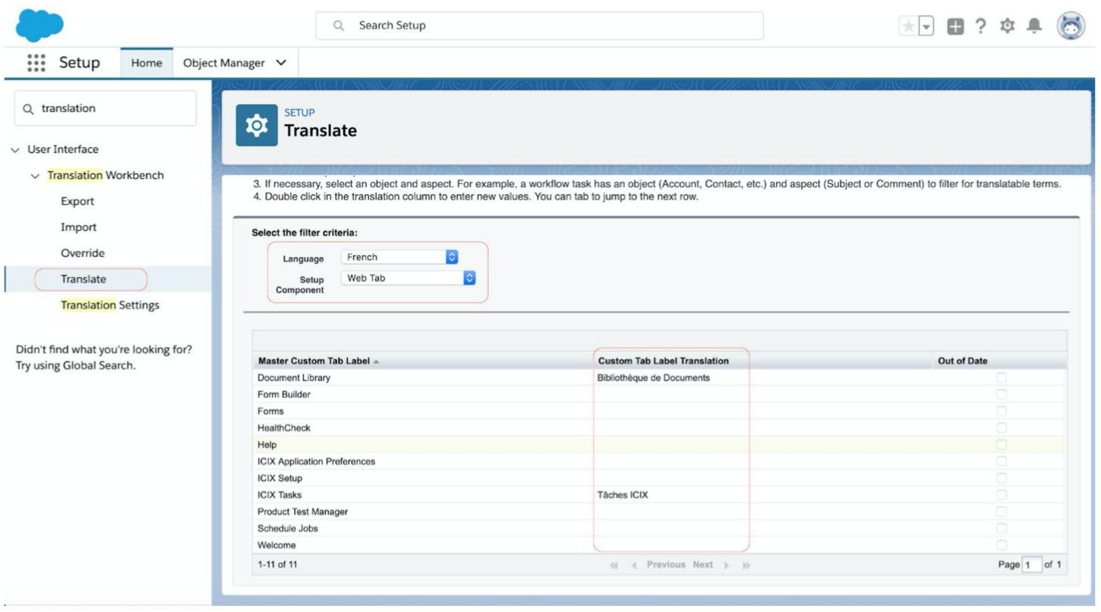
- **Language:** As at the Custom Labels window, this is the language we will translate the picklist values to. The languages displayed are limited to the languages selected in the **Translation Workbench**.
- **Setup Component:** Displays a list of possible components to translate like lookup filter, custom field, picklist, web tab, and many others. In this case, we will select **Picklist Value** which will make the second section of the **Translate** page refresh and display a list of **Master Picklist Value Label**.
- **Object**: When a component is selected, a list of possible objects from the previous step are displayed. Since we selected a picklist, all options will be existing and active picklists to translate.

In the lower section of the **Translate** window, click on the **"+".** All existing values in the picklist are displayed. There is a column named **Picklist Value Label Translation**. By double-clicking the value you need to translate, or by clicking on the pencil icon, you are able to enter a translation. Once you are happy with the translation(s), remember to click the **Save** button.

How to Translate Tabs in Salesforce

VisualForce Tab or Web Tab

In order to translate a tab we need to go to, **Setup** → in **Quick Find** search for “***translate*"**:



3. If necessary, select an object and aspect. For example, a workflow task has an object (Account, Contact, etc.) and aspect (Subject or Comment) to filter for translatable terms.
4. Double click in the translation column to enter new values. You can tab to jump to the next row.

Select the filter criteria:

Language: French
Setup Component: Web Tab

Master Custom Tab Label	Custom Tab Label Translation	Out of Date
Document Library	Bibliothèque de Documents	☐
Form Builder		☐
Forms		☐
HealthCheck		☐
Help		☐
ICIX Application Preferences		☐
ICIX Setup		☐
ICIX Tasks	Tâches ICIX	☐
Product Test Manager		☐
Schedule Jobs		☐
Welcome		☐

1-11 of 11 · Previous Next · Page 1 of 1

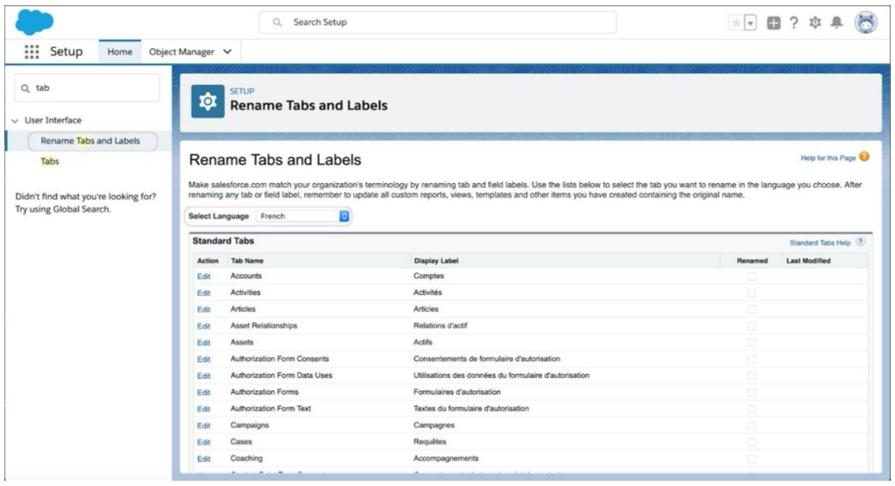
The Translate window is divided into two sections. The first one, on top, displays two fields that need to be described in order to carry out the required translations.

- **Language:** As with the Custom Labels window, this is the language we will be translating the tab to. The languages displayed are limited to the languages selected in the **Translation Workbench**.
- **Setup Component:** Displays a list of possible components to translate, like lookup filter, custom field, picklist, web tab, among many others. In this case we will select the **"Web tab"** which will make the second section of the **Translate** page refresh and display a list of **Master Custom Tab Label**.

In the second section of the **Translate** window, all web tabs are displayed. There is a column named **Custom Tab Label Translation.** By double-clicking the tab you need to translate, or by clicking on the pencil icon, you are able to enter a translation. Once you are ready with the translation(s), remember to click the **Save** button.

Custom Tab

In order to rename a tab, we need to go to **Setup** → in **Quick Find** search for **"*Rename Tabs and Labels*"**:



Rename Tabs and Labels

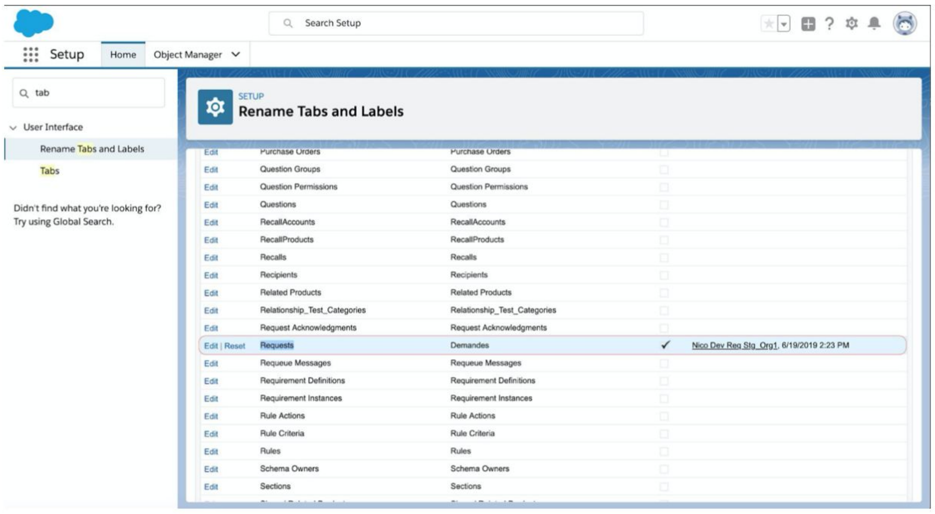
Make salesforce.com match your organization's terminology by renaming tab and field labels. Use the lists below to select the tab you want to rename in the language you choose. After renaming any tab or field label, remember to update all custom reports, views, templates and other items you have created containing the original name.

Select Language: French

Standard Tabs

Action	Tab Name	Display Label	Renamed	Last Modified
Edit	Accounts	Comptes	☐	
Edit	Activities	Activités	☐	
Edit	Articles	Articles	☐	
Edit	Asset Relationships	Relations d'actif	☐	
Edit	Assets	Actifs	☐	
Edit	Authorization Form Consents	Consentements de formulaire d'autorisation	☐	
Edit	Authorization Form Data Uses	Utilisations des données du formulaire d'autorisation	☐	
Edit	Authorization Forms	Formulaires d'autorisation	☐	
Edit	Authorization Form Text	Textes du formulaire d'autorisation	☐	
Edit	Campaigns	Campagnes	☐	
Edit	Cases	Requêtes	☐	
Edit	Coaching	Accompagnements	☐	

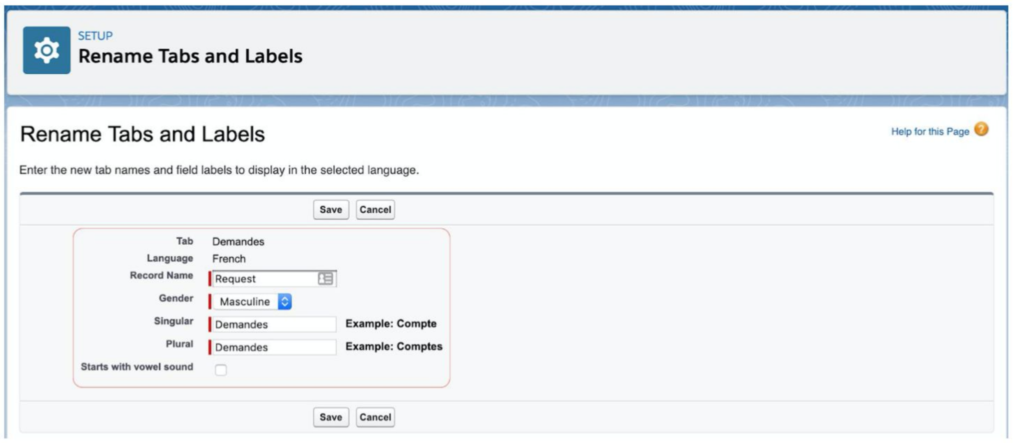
At the **Rename Tabs and Labels** page, select the language to translate the tab to by clicking on **Select Language** drop list.



Edit	Purchase Orders	Purchase Orders	☐	
Edit	Question Groups	Question Groups	☐	
Edit	Question Permissions	Question Permissions	☐	
Edit	Questions	Questions	☐	
Edit	RecallAccounts	RecallAccounts	☐	
Edit	RecallProducts	RecallProducts	☐	
Edit	Recalls	Recalls	☐	
Edit	Recipients	Recipients	☐	
Edit	Related Products	Related Products	☐	
Edit	Relationship_Test_Categories	Relationship_Test_Categories	☐	
Edit	Request Acknowledgments	Request Acknowledgments	☐	
Edit | Reset	Requests	Demandes	✓	Nico Dev Req Stg_Org1, 6/19/2019 2:23 PM
Edit	Requeue Messages	Requeue Messages	☐	
Edit	Requirement Definitions	Requirement Definitions	☐	
Edit	Requirement Instances	Requirement Instances	☐	
Edit	Rule Actions	Rule Actions	☐	
Edit	Rule Criteria	Rule Criteria	☐	
Edit	Rules	Rules	☐	
Edit	Schema Owners	Schema Owners	☐	
Edit	Sections	Sections	☐	

A translated custom tab

Once you have selected a language at the **Select Language** drop list, click on **Edit,** found in the **Action** column for the tab we would like to translate. This will open the **Rename Tabs and Labels** page.



Rename Tabs and Labels

Enter the new tab names and field labels to display in the selected language.

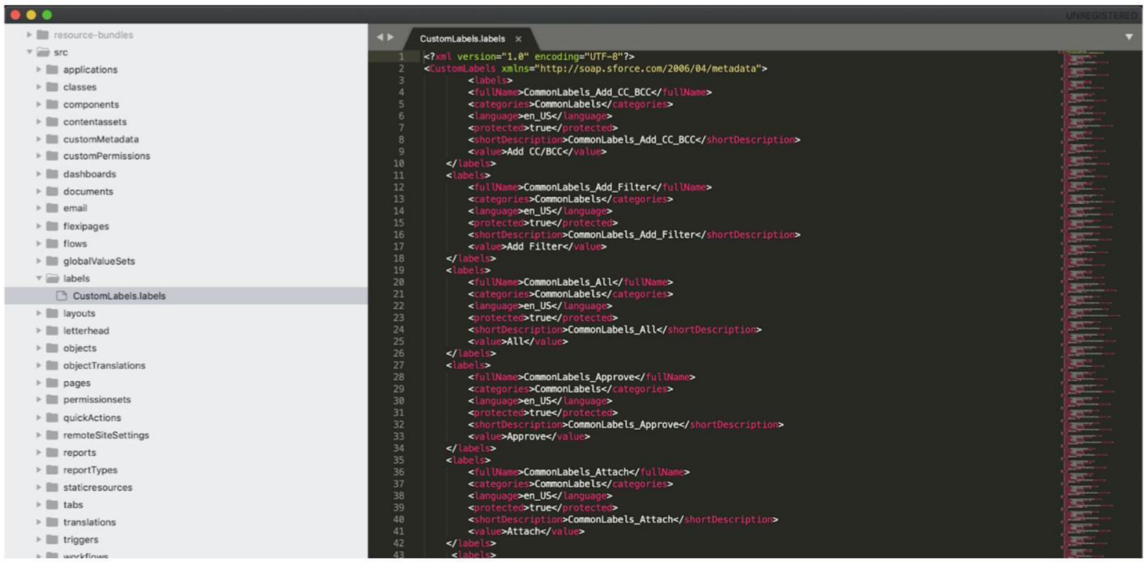
Tab	Demandes	
Language	French	
Record Name	Request	
Gender	Masculine	
Singular	Demandes	Example: Compte
Plural	Demandes	Example: Comptes
Starts with vowel sound	☐	

- **Tab:** Displays the text of the tab on the selected language at **"Select Language"** drop list. This field is disabled.
- **Language:** Displays the language selected in the **"Select Language"** drop list. This is the language to which the User will translate the **"Record Name"**. This field is disabled.
- **Record Name:** The text of the tab displayed by default.
- **Gender:** If the text is male or female.
- **Singular:** Indicate how the text needs to display in singular form.
- **Plural:** Indicate how the text needs to display in singular form.

How to Include Labels on the Back-End

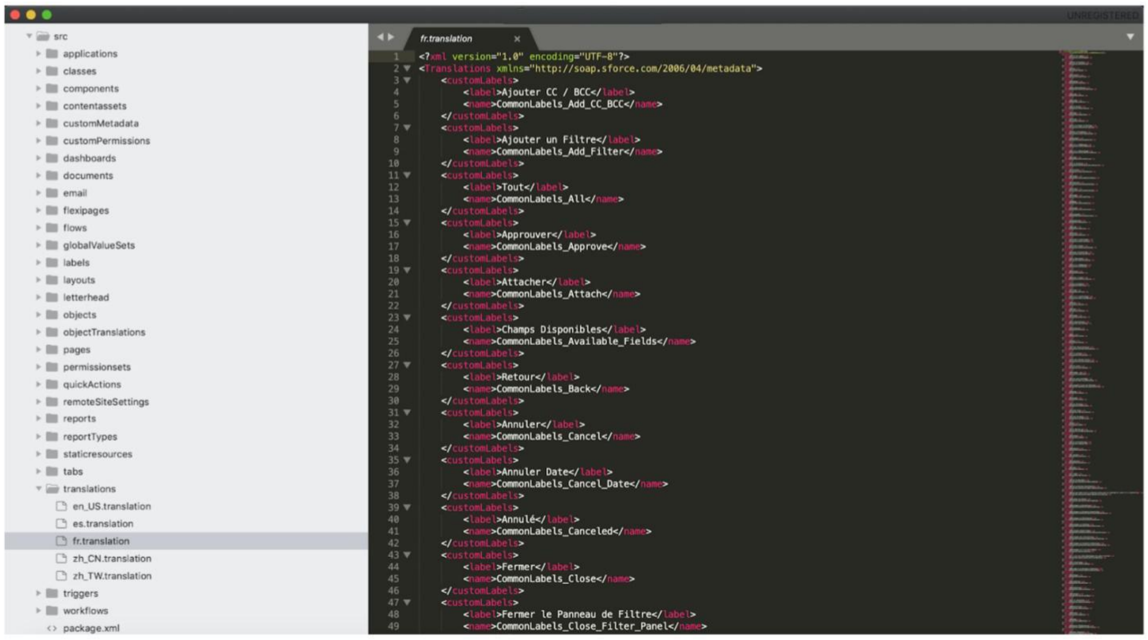
Custom Label Metadata

Custom label files and related metadata must be included alphabetically in ***src/labels***.



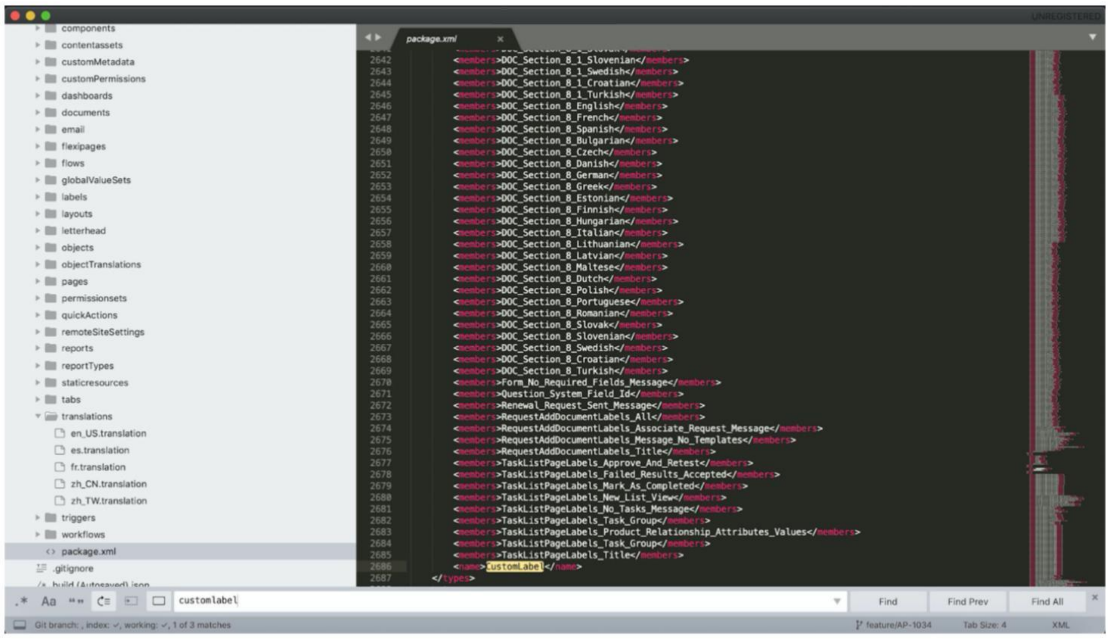
```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomLabels xmlns="http://soap.sforce.com/2006/04/metadata">
        <labels>
        <fullName>CommonLabels_Add_CC_BCC</fullName>
        <categories>CommonLabels</categories>
        <language>en_US</language>
        <protected>true</protected>
        <shortDescription>CommonLabels_Add_CC_BCC</shortDescription>
        <value>Add CC/BCC</value>
    </labels>
    <labels>
        <fullName>CommonLabels_Add_Filter</fullName>
        <categories>CommonLabels</categories>
        <language>en_US</language>
        <protected>true</protected>
        <shortDescription>CommonLabels_Add_Filter</shortDescription>
        <value>Add Filter</value>
    </labels>
    <labels>
        <fullName>CommonLabels_All</fullName>
        <categories>CommonLabels</categories>
        <language>en_US</language>
        <protected>true</protected>
        <shortDescription>CommonLabels_All</shortDescription>
        <value>All</value>
    </labels>
    <labels>
        <fullName>CommonLabels_Approve</fullName>
        <categories>CommonLabels</categories>
        <language>en_US</language>
        <protected>true</protected>
        <shortDescription>CommonLabels_Approve</shortDescription>
        <value>Approve</value>
    </labels>
    <labels>
        <fullName>CommonLabels_Attach</fullName>
        <categories>CommonLabels</categories>
        <language>en_US</language>
        <protected>true</protected>
        <shortDescription>CommonLabels_Attach</shortDescription>
        <value>Attach</value>
    </labels>
     <labels>
```

Custom label translations have to be included, ordered alphabetically in ***src/translations***.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<Translations xmlns="http://soap.sforce.com/2006/04/metadata">
    <customLabels>
        <label>Ajouter CC / BCC</label>
        <name>CommonLabels_Add_CC_BCC</name>
    </customLabels>
    <customLabels>
        <label>Ajouter un Filtre</label>
        <name>CommonLabels_Add_Filter</name>
    </customLabels>
    <customLabels>
        <label>Tout</label>
        <name>CommonLabels_All</name>
    </customLabels>
    <customLabels>
        <label>Approuver</label>
        <name>CommonLabels_Approve</name>
    </customLabels>
    <customLabels>
        <label>Attacher</label>
        <name>CommonLabels_Attach</name>
    </customLabels>
    <customLabels>
        <label>Champs Disponibles</label>
        <name>CommonLabels_Available_Fields</name>
    </customLabels>
    <customLabels>
        <label>Retour</label>
        <name>CommonLabels_Back</name>
    </customLabels>
    <customLabels>
        <label>Annuler</label>
        <name>CommonLabels_Cancel</name>
    </customLabels>
    <customLabels>
        <label>Annuler Date</label>
        <name>CommonLabels_Cancel_Date</name>
    </customLabels>
    <customLabels>
        <label>Annulé</label>
        <name>CommonLabels_Canceled</name>
    </customLabels>
    <customLabels>
        <label>Fermer</label>
        <name>CommonLabels_Close</name>
    </customLabels>
    <customLabels>
        <label>Fermer le Panneau de Filtre</label>
        <name>CommonLabels_Close_Filter_Panel</name>
```

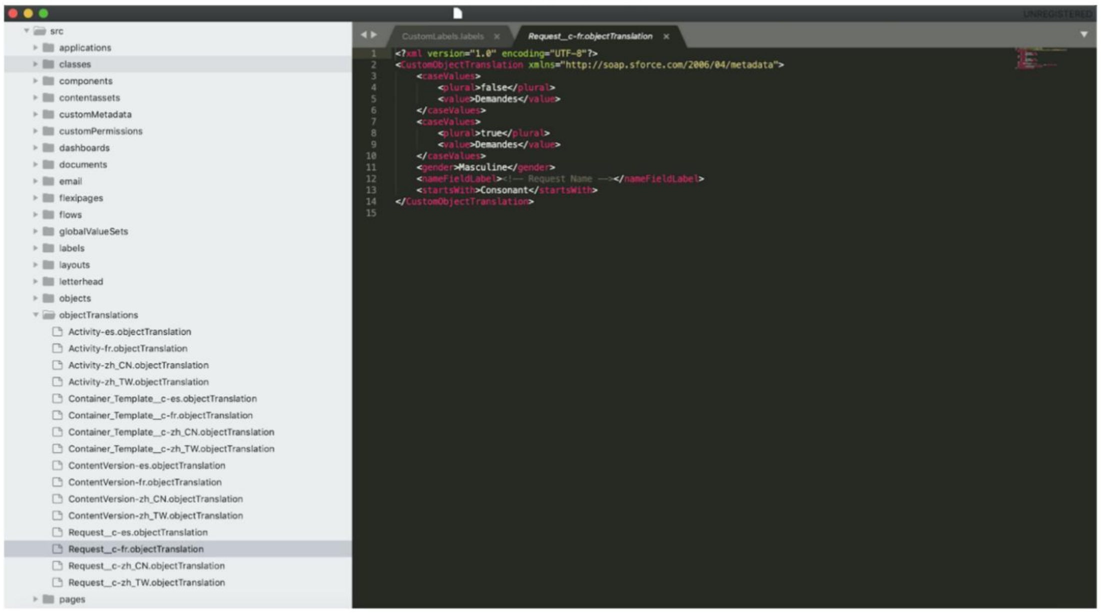
In *src/translations*, a file for each language is included with a different prefix with which it can be identified and differentiated from the rest of the existing languages. In each file you will find between **<customLabels></customLabels>** the text of the translated label for this particular language.
At the same time, each label must be included in the ***package.xml*** manifest, referenced by its API name, in alphabetical order.



```xml
            <members>DOC_Section_8_1_Slovenian</members>
            <members>DOC_Section_8_1_Swedish</members>
            <members>DOC_Section_8_1_Croatian</members>
            <members>DOC_Section_8_1_Turkish</members>
            <members>DOC_Section_8_English</members>
            <members>DOC_Section_8_French</members>
            <members>DOC_Section_8_Spanish</members>
            <members>DOC_Section_8_Bulgarian</members>
            <members>DOC_Section_8_Czech</members>
            <members>DOC_Section_8_Danish</members>
            <members>DOC_Section_8_German</members>
            <members>DOC_Section_8_Greek</members>
            <members>DOC_Section_8_Estonian</members>
            <members>DOC_Section_8_Finnish</members>
            <members>DOC_Section_8_Hungarian</members>
            <members>DOC_Section_8_Italian</members>
            <members>DOC_Section_8_Lithuanian</members>
            <members>DOC_Section_8_Latvian</members>
            <members>DOC_Section_8_Maltese</members>
            <members>DOC_Section_8_Dutch</members>
            <members>DOC_Section_8_Polish</members>
            <members>DOC_Section_8_Portuguese</members>
            <members>DOC_Section_8_Romanian</members>
            <members>DOC_Section_8_Slovak</members>
            <members>DOC_Section_8_Slovenian</members>
            <members>DOC_Section_8_Swedish</members>
            <members>DOC_Section_8_Croatian</members>
            <members>DOC_Section_8_Turkish</members>
            <members>Form_No_Required_Fields_Message</members>
            <members>Question_System_Field_Id</members>
            <members>Renewal_Request_Sent_Message</members>
            <members>RequestAddDocumentLabels_All</members>
            <members>RequestAddDocumentLabels_Associate_Request_Message</members>
            <members>RequestAddDocumentLabels_Message_No_Templates</members>
            <members>RequestAddDocumentLabels_Title</members>
            <members>TaskListPageLabels_Approve_And_Retest</members>
            <members>TaskListPageLabels_Failed_Results_Accepted</members>
            <members>TaskListPageLabels_Mark_As_Completed</members>
            <members>TaskListPageLabels_New_List_View</members>
            <members>TaskListPageLabels_No_Tasks_Message</members>
            <members>TaskListPageLabels_Task_Group</members>
            <members>TaskListPageLabels_Product_Relationship_Attributes_Values</members>
            <members>TaskListPageLabels_Task_Group</members>
            <members>TaskListPageLabels_Title</members>
            <name>CustomLabel</name>
        </types>
```

Web Tabs metadata

Web tab files and related metadata must be included in *src/objectTranslations*.

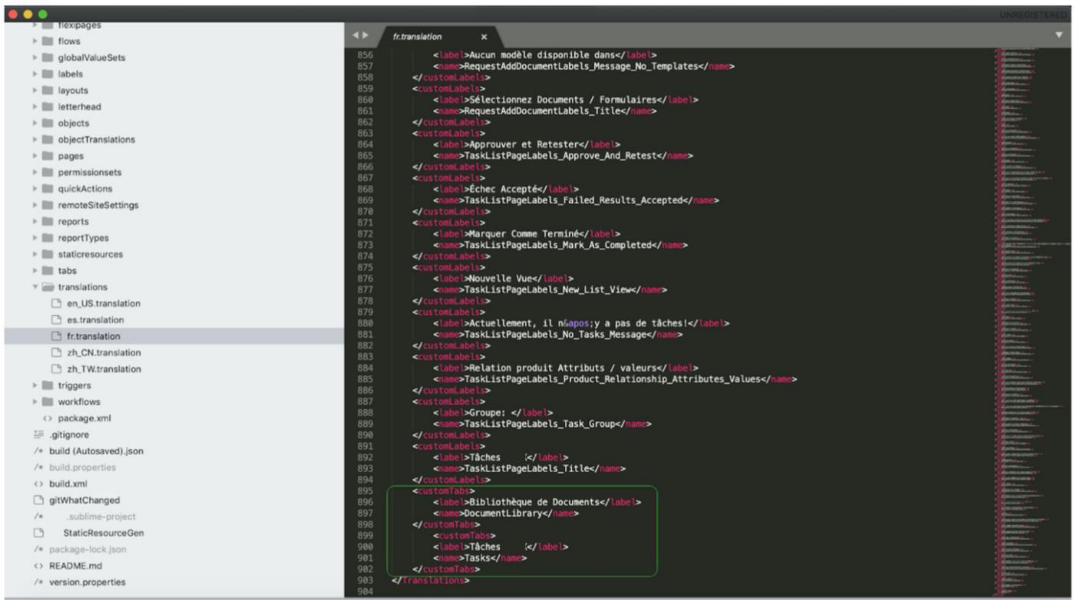

src
applications
classes
components
contentassets
customMetadata
customPermissions
dashboards
documents
email
flexipages
flows
globalValueSets
labels
layouts
letterhead
objects
objectTranslations
Activity-es.objectTranslation
Activity-fr.objectTranslation
Activity-zh_CN.objectTranslation
Activity-zh_TW.objectTranslation
Container_Template__c-es.objectTranslation
Container_Template__c-fr.objectTranslation
Container_Template__c-zh_CN.objectTranslation
Container_Template__c-zh_TW.objectTranslation
ContentVersion-es.objectTranslation
ContentVersion-fr.objectTranslation
ContentVersion-zh_CN.objectTranslation
ContentVersion-zh_TW.objectTranslation
Request__c-es.objectTranslation
Request__c-fr.objectTranslation
Request__c-zh_CN.objectTranslation
Request__c-zh_TW.objectTranslation
pages
CustomLabels.labels
Request__c-fr.objectTranslation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseValues>
        <plural>false</plural>
        <value>Demandes</value>
    </caseValues>
    <caseValues>
        <plural>true</plural>
        <value>Demandes</value>
    </caseValues>
    <gender>Masculine</gender>
    <nameFieldLabel><!-- Request Name --></nameFieldLabel>
    <startsWith>Consonant</startsWith>
</CustomObjectTranslation>
```

In *src/objectTranslations* a file for each required language and each object must be included. In this file translation for Web tabs and for other objects like the picklist must be included.

Custom Tabs metadata

Custom tab files and related metadata must be included in *src/objectTranslations*.

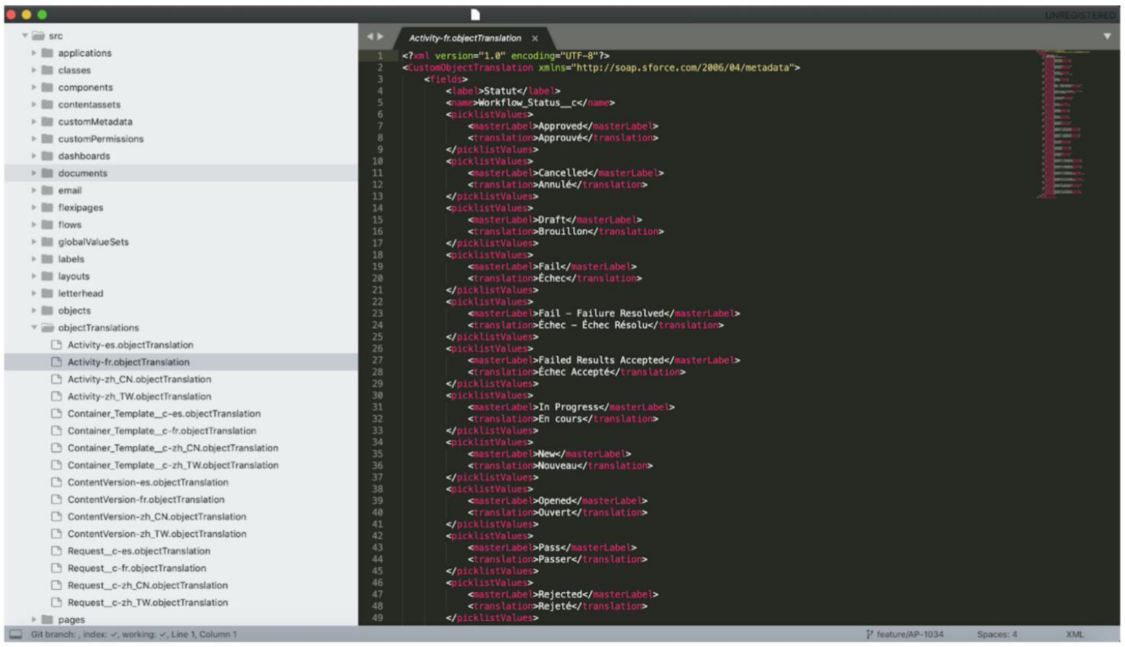


```xml
            <label>Aucun modèle disponible dans</label>
            <name>RequestAddDocumentLabels_Message_No_Templates</name>
        </customLabels>
        <customLabels>
            <label>Sélectionnez Documents / Formulaires</label>
            <name>RequestAddDocumentLabels_Title</name>
        </customLabels>
        <customLabels>
            <label>Approuver et Retester</label>
            <name>TaskListPageLabels_Approve_And_Retest</name>
        </customLabels>
        <customLabels>
            <label>Échec Accepté</label>
            <name>TaskListPageLabels_Failed_Results_Accepted</name>
        </customLabels>
        <customLabels>
            <label>Marquer Comme Terminé</label>
            <name>TaskListPageLabels_Mark_As_Completed</name>
        </customLabels>
        <customLabels>
            <label>Nouvelle Vue</label>
            <name>TaskListPageLabels_New_List_View</name>
        </customLabels>
        <customLabels>
            <label>Actuellement, il n&apos;y a pas de tâches!</label>
            <name>TaskListPageLabels_No_Tasks_Message</name>
        </customLabels>
        <customLabels>
            <label>Relation produit Attributs / valeurs</label>
            <name>TaskListPageLabels_Product_Relationship_Attributes_Values</name>
        </customLabels>
        <customLabels>
            <label>Groupe: </label>
            <name>TaskListPageLabels_Task_Group</name>
        </customLabels>
        <customLabels>
            <label>Tâches     </label>
            <name>TaskListPageLabels_Title</name>
        </customLabels>
        <customTabs>
            <label>Bibliothèque de Documents</label>
            <name>DocumentLibrary</name>
        </customTabs>
            <customTabs>
            <label>Tâches     </label>
            <name>Tasks</name>
        </customTabs>
    </Translations>
```

In *src/translations* a file for each language is included with a different prefix with which it can be identified and differentiated from the rest of the existing languages. In each file you will find between **<customTabs></customTabs>** the text of the translated tab for this particular language.

Picklist metadata

Picklist files and related metadata must be included in *src/objectTranslations*.

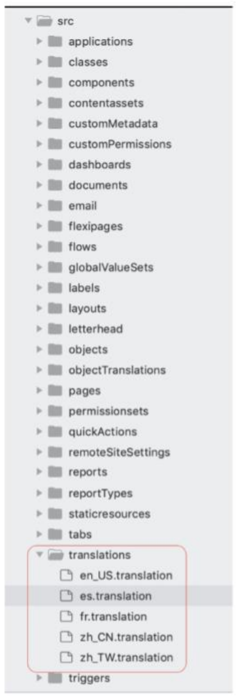

src
applications
classes
components
contentassets
customMetadata
customPermissions
dashboards
documents
email
flexipages
flows
globalValueSets
labels
layouts
letterhead
objects
objectTranslations
Activity-es.objectTranslation
Activity-fr.objectTranslation
Activity-zh_CN.objectTranslation
Activity-zh_TW.objectTranslation
Container_Template__c-es.objectTranslation
Container_Template__c-fr.objectTranslation
Container_Template__c-zh_CN.objectTranslation
Container_Template__c-zh_TW.objectTranslation
ContentVersion-es.objectTranslation
ContentVersion-fr.objectTranslation
ContentVersion-zh_CN.objectTranslation
ContentVersion-zh_TW.objectTranslation
Request__c-es.objectTranslation
Request__c-fr.objectTranslation
Request__c-zh_CN.objectTranslation
Request__c-zh_TW.objectTranslation
pages
Activity-fr.objectTranslation
UNREGISTERED
Git branch: , index: ✓, working: ✓, Line 1, Column 1
feature/AP-1034
Spaces: 4
XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <label>Statut</label>
        <name>Workflow_Status__c</name>
        <picklistValues>
            <masterLabel>Approved</masterLabel>
            <translation>Approuvé</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Cancelled</masterLabel>
            <translation>Annulé</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Draft</masterLabel>
            <translation>Brouillon</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Fail</masterLabel>
            <translation>Échec</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Fail - Failure Resolved</masterLabel>
            <translation>Échec - Échec Résolu</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Failed Results Accepted</masterLabel>
            <translation>Échec Accepté</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>In Progress</masterLabel>
            <translation>En cours</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>New</masterLabel>
            <translation>Nouveau</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Opened</masterLabel>
            <translation>Ouvert</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Pass</masterLabel>
            <translation>Passer</translation>
        </picklistValues>
        <picklistValues>
            <masterLabel>Rejected</masterLabel>
            <translation>Rejeté</translation>
        </picklistValues>
```

In *src/objectTranslations* a file for each required language and each object must be included. In this file translation for picklist values and for other objects must be included.

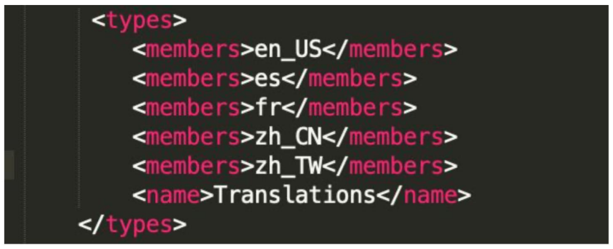
Languages

Language files and related metadata must be included in ***src/objectTranslations***.



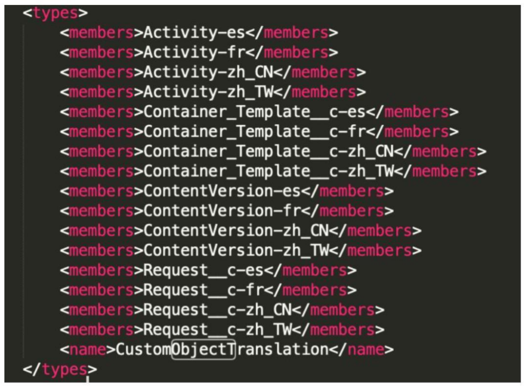
```
src
  applications
  classes
  components
  contentassets
  customMetadata
  customPermissions
  dashboards
  documents
  email
  flexipages
  flows
  globalValueSets
  labels
  layouts
  letterhead
  objects
  objectTranslations
  pages
  permissionsets
  quickActions
  remoteSiteSettings
  reports
  reportTypes
  staticresources
  tabs
  translations
    en_US.translation
    es.translation
    fr.translation
    zh_CN.translation
    zh_TW.translation
  triggers
```

In src/translations a file for each language is included with a different prefix with which it can be identified and differentiated from the rest of the existing languages. In each file you will find the translations for each particular language.



```
    <types>
        <members>en_US</members>
        <members>es</members>
        <members>fr</members>
        <members>zh_CN</members>
        <members>zh_TW</members>
        <name>Translations</name>
    </types>
```

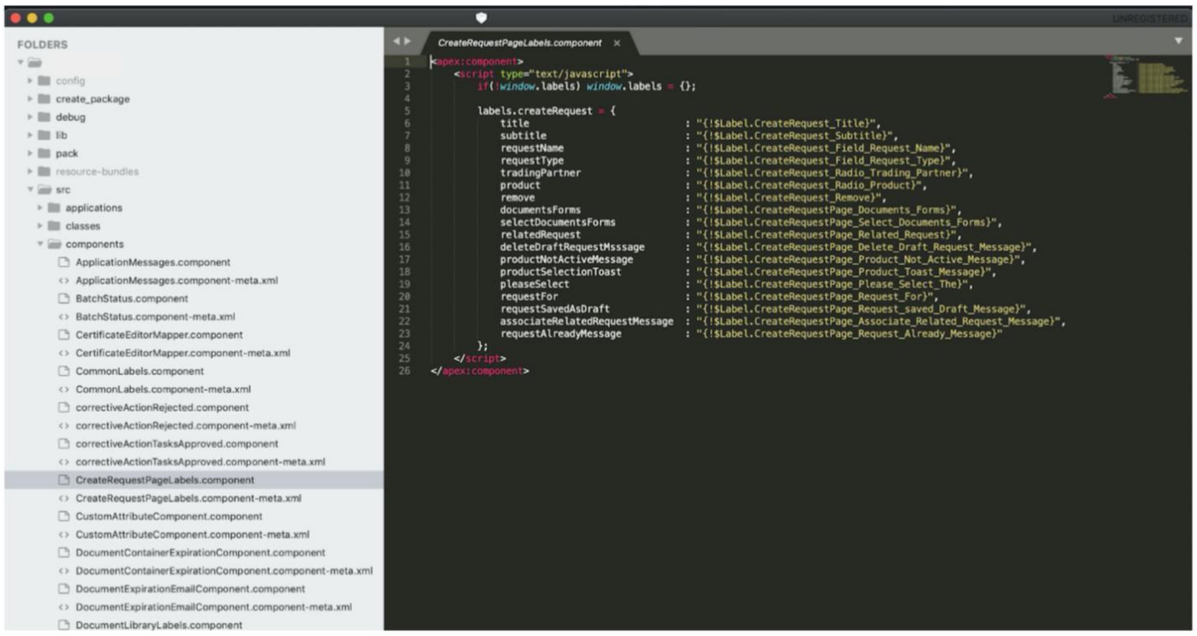
At the same time, each translation and objectTranslations must be included in the ***package.xml*** manifest, referenced by its API name and maintaining the alphabetical order.



```
<types>
    <members>Activity-es</members>
    <members>Activity-fr</members>
    <members>Activity-zh_CN</members>
    <members>Activity-zh_TW</members>
    <members>Container_Template__c-es</members>
    <members>Container_Template__c-fr</members>
    <members>Container_Template__c-zh_CN</members>
    <members>Container_Template__c-zh_TW</members>
    <members>ContentVersion-es</members>
    <members>ContentVersion-fr</members>
    <members>ContentVersion-zh_CN</members>
    <members>ContentVersion-zh_TW</members>
    <members>Request__c-es</members>
    <members>Request__c-fr</members>
    <members>Request__c-zh_CN</members>
    <members>Request__c-zh_TW</members>
    <name>CustomObjectTranslation</name>
</types>
```

Working at Code-Level with the Different Components Created

In order to translate a page (Visualforce page) we must apply the following pattern or predefined standard for creating a Visualforce component (Apex) to group all labels related to the page.
The created component must be included under ***src/components***.


FOLDERS
config
create_package
debug
lib
pack
resource-bundles
src
applications
classes
components
ApplicationMessages.component
ApplicationMessages.component-meta.xml
BatchStatus.component
BatchStatus.component-meta.xml
CertificateEditorMapper.component
CertificateEditorMapper.component-meta.xml
CommonLabels.component
CommonLabels.component-meta.xml
correctiveActionRejected.component
correctiveActionRejected.component-meta.xml
correctiveActionTasksApproved.component
correctiveActionTasksApproved.component-meta.xml
CreateRequestPageLabels.component
CreateRequestPageLabels.component-meta.xml
CustomAttributeComponent.component
CustomAttributeComponent.component-meta.xml
DocumentContainerExpirationComponent.component
DocumentContainerExpirationComponent.component-meta.xml
DocumentExpirationEmailComponent.component
DocumentExpirationEmailComponent.component-meta.xml
DocumentLibraryLabels.component
CreateRequestPageLabels.component

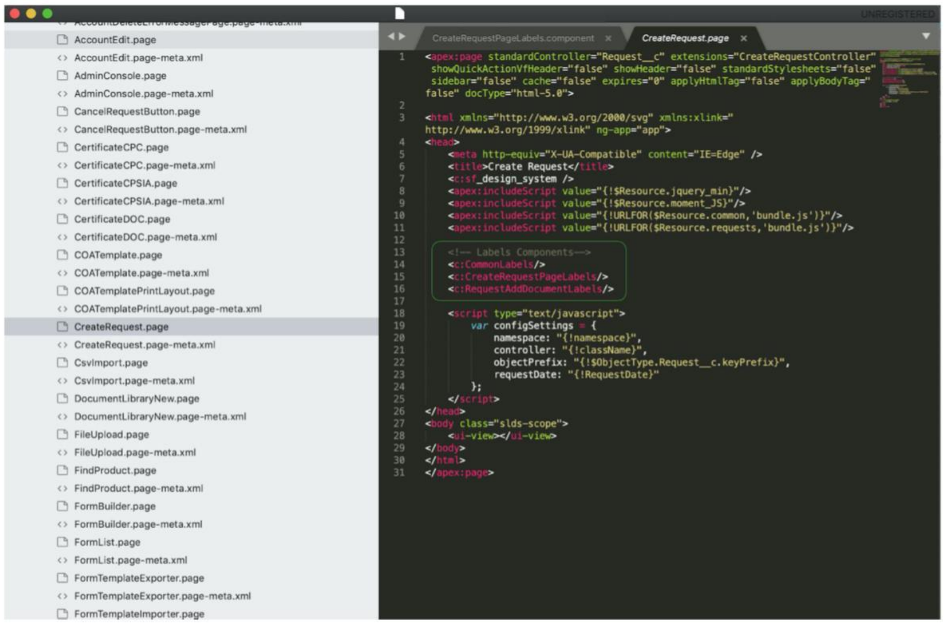
```
<apex:component>
    <script type="text/javascript">
        if(!window.labels) window.labels = {};

        labels.createRequest = {
            title                          : "{!$Label.CreateRequest_Title}",
            subtitle                       : "{!$Label.CreateRequest_Subtitle}",
            requestName                    : "{!$Label.CreateRequest_Field_Request_Name}",
            requestType                    : "{!$Label.CreateRequest_Field_Request_Type}",
            tradingPartner                 : "{!$Label.CreateRequest_Radio_Trading_Partner}",
            product                        : "{!$Label.CreateRequest_Radio_Product}",
            remove                         : "{!$Label.CreateRequest_Remove}",
            documentsForms                 : "{!$Label.CreateRequestPage_Documents_Forms}",
            selectDocumentsForms           : "{!$Label.CreateRequestPage_Select_Documents_Forms}",
            relatedRequest                 : "{!$Label.CreateRequestPage_Related_Request}",
            deleteDraftRequestMsssage      : "{!$Label.CreateRequestPage_Delete_Draft_Request_Message}",
            productNotActiveMessage        : "{!$Label.CreateRequestPage_Product_Not_Active_Message}",
            productSelectionToast          : "{!$Label.CreateRequestPage_Product_Toast_Message}",
            pleaseSelect                   : "{!$Label.CreateRequestPage_Please_Select_The}",
            requestFor                     : "{!$Label.CreateRequestPage_Request_For}",
            requestSavedAsDraft            : "{!$Label.CreateRequestPage_Request_saved_Draft_Message}",
            associateRelatedRequestMessage : "{!$Label.CreateRequestPage_Associate_Related_Request_Message}",
            requestAlreadyMessage          : "{!$Label.CreateRequestPage_Request_Already_Message}"
        };
    </script>
</apex:component>
```

In this file all custom labels will be included which will be used exclusively inside the page to translate. In the previous screenshot we can see a component created for **CreateRequestPage** where the name of the component is **CreateRequestPageLabels**.

This component naming convention must be followed. This helps to easily identify its purpose and also helps keep coherence throughout the code at the time of developing.

A property needs to be created inside the JavaScript global variable named **"labels"** with the name of the page to translate being of type Object **"labels.createRequest = { }"**. Inside this new property (createRequest), which belongs to the Label object, we need to include all custom labels previously created in the org with a name to identify them at code level, eg:

```
labels.createRequest = {
                    title           : "{!$Label.CreateRequest_Title}",
                    subtitle        : "{!$Label.CreateRequest_Subtitle}
};
```


AccountEdit.page
AccountEdit.page-meta.xml
AdminConsole.page
AdminConsole.page-meta.xml
CancelRequestButton.page
CancelRequestButton.page-meta.xml
CertificateCPC.page
CertificateCPC.page-meta.xml
CertificateCPSIA.page
CertificateCPSIA.page-meta.xml
CertificateDOC.page
CertificateDOC.page-meta.xml
COATemplate.page
COATemplate.page-meta.xml
COATemplatePrintLayout.page
COATemplatePrintLayout.page-meta.xml
CreateRequest.page
CreateRequest.page-meta.xml
CsvImport.page
CsvImport.page-meta.xml
DocumentLibraryNew.page
DocumentLibraryNew.page-meta.xml
FileUpload.page
FileUpload.page-meta.xml
FindProduct.page
FindProduct.page-meta.xml
FormBuilder.page
FormBuilder.page-meta.xml
FormList.page
FormList.page-meta.xml
FormTemplateExporter.page
FormTemplateExporter.page-meta.xml
FormTemplateImporter.page
CreateRequestPageLabels.component
CreateRequest.page

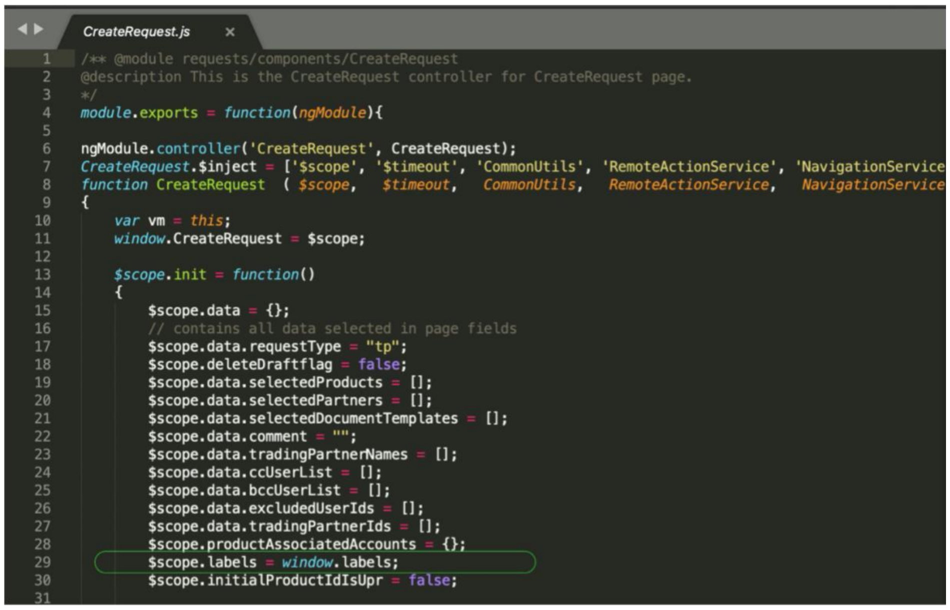
```
<apex:page standardController="Request__c" extensions="CreateRequestController"
 showQuickActionVfHeader="false" showHeader="false" standardStylesheets="false"
 sidebar="false" cache="false" expires="0" applyHtmlTag="false" applyBodyTag="
false" docType="html-5.0">

<html xmlns="http://www.w3.org/2000/svg" xmlns:xlink="
http://www.w3.org/1999/xlink" ng-app="app">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
    <title>Create Request</title>
    <c:sf_design_system />
    <apex:includeScript value="{!$Resource.jquery_min}"/>
    <apex:includeScript value="{!$Resource.moment_JS}"/>
    <apex:includeScript value="{!URLFOR($Resource.common,'bundle.js')}"/>
    <apex:includeScript value="{!URLFOR($Resource.requests,'bundle.js')}"/>

    <!-- Labels Components-->
    <c:CommonLabels/>
    <c:CreateRequestPageLabels/>
    <c:RequestAddDocumentLabels/>

    <script type="text/javascript">
        var configSettings = {
            namespace: "{!namespace}",
            controller: "{!className}",
            objectPrefix: "{!$ObjectType.Request__c.keyPrefix}",
            requestDate: "{!RequestDate}"
        };
    </script>
</head>
<body class="slds-scope">
    <ui-view></ui-view>
</body>
</html>
</apex:page>
```

These components must be included from the page we need to translate as shown in the previous screenshot, **<c:CreateRequestPageLabels/>**.

NOTE: Those labels which could be considered as regular use (which could be applied in more than one page), must be included in the Apex component **"CommonLabels.component"** and always be included in each page we need to translate as shown in the previous screenshot, **<c:CommonLabels/>**. In cases where AngularJS sub-components are considered large and at the same time are reused throughout the application, we might need to create another Apex component to group the labels (also as shown in the previous screenshot).

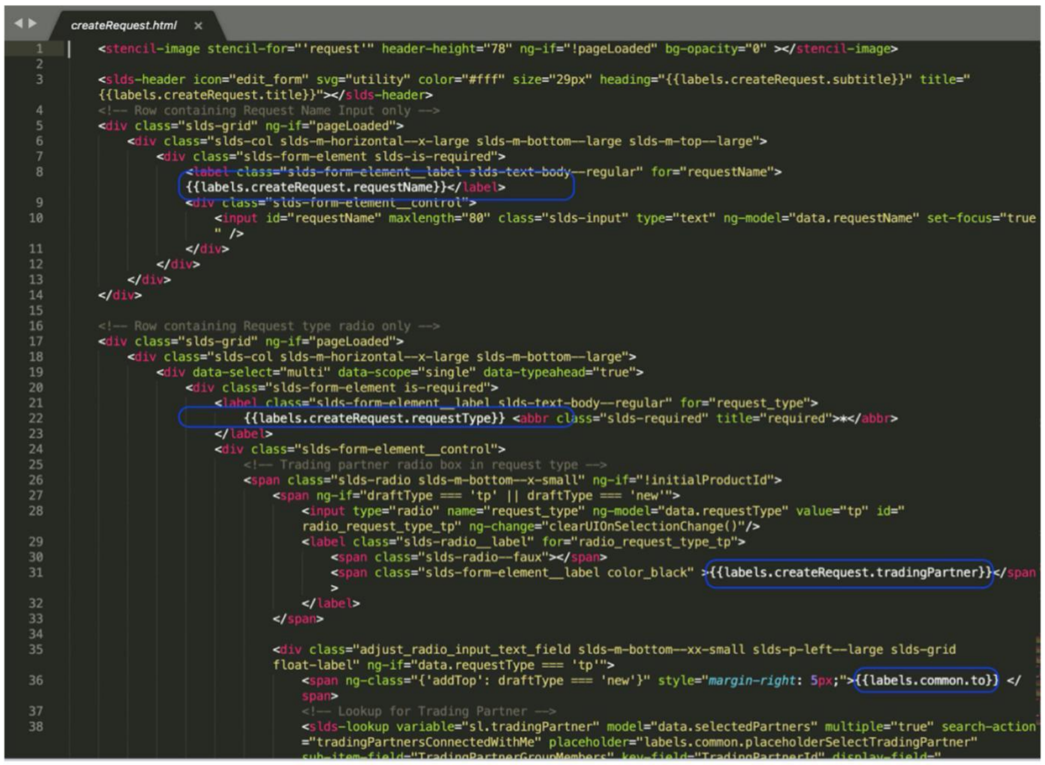
How to reference Custom Labels from AngularJS

In order to reference labels from AngularJS code or HTML; for each page to translate, we must include in the scope of the embedded Angular application those labels which are found on the Apex components.



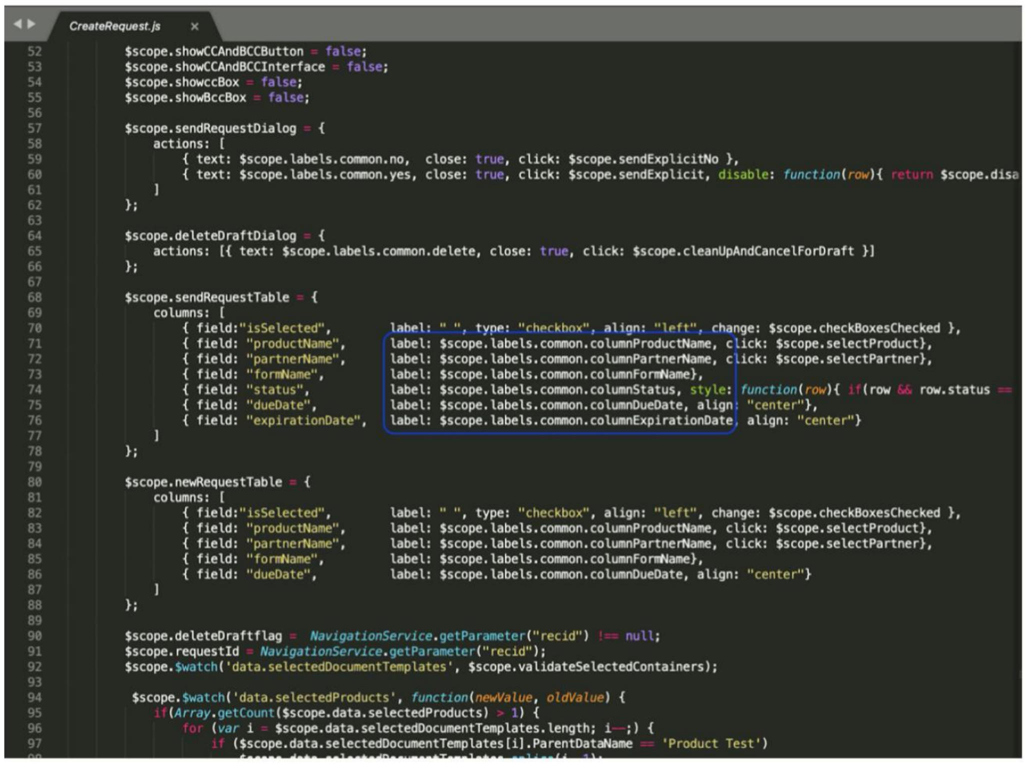
```
/** @module requests/components/CreateRequest
@description This is the CreateRequest controller for CreateRequest page.
*/
module.exports = function(ngModule){

ngModule.controller('CreateRequest', CreateRequest);
CreateRequest.$inject = ['$scope', '$timeout', 'CommonUtils', 'RemoteActionService', 'NavigationService
function CreateRequest ( $scope,  $timeout,  CommonUtils,  RemoteActionService,  NavigationService
{
    var vm = this;
    window.CreateRequest = $scope;

    $scope.init = function()
    {
        $scope.data = {};
        // contains all data selected in page fields
        $scope.data.requestType = "tp";
        $scope.deleteDraftflag = false;
        $scope.data.selectedProducts = [];
        $scope.data.selectedPartners = [];
        $scope.data.selectedDocumentTemplates = [];
        $scope.data.comment = "";
        $scope.data.tradingPartnerNames = [];
        $scope.data.ccUserList = [];
        $scope.data.bccUserList = [];
        $scope.data.excludedUserIds = [];
        $scope.data.tradingPartnerIds = [];
        $scope.productAssociatedAccounts = {};
        $scope.labels = window.labels;
        $scope.initialProductIdIsUpr = false;
```

Once we include the labels into the scope we can reference them from **HTML** or **AngularJS** in the following way(s):



```html
<stencil-image stencil-for="'request'" header-height="78" ng-if="!pageLoaded" bg-opacity="0" ></stencil-image>

<slds-header icon="edit_form" svg="utility" color="#fff" size="29px" heading="{{labels.createRequest.subtitle}}" title="
{{labels.createRequest.title}}"></slds-header>
<!-- Row containing Request Name Input only -->
<div class="slds-grid" ng-if="pageLoaded">
    <div class="slds-col slds-m-horizontal--x-large slds-m-bottom--large slds-m-top--large">
        <div class="slds-form-element slds-is-required">
            <label class="slds-form-element__label slds-text-body--regular" for="requestName">
            {{labels.createRequest.requestName}}</label>
            <div class="slds-form-element__control">
                <input id="requestName" maxlength="80" class="slds-input" type="text" ng-model="data.requestName" set-focus="true
                " />
            </div>
        </div>
    </div>
</div>

<!-- Row containing Request type radio only -->
<div class="slds-grid" ng-if="pageLoaded">
    <div class="slds-col slds-m-horizontal--x-large slds-m-bottom--large">
        <div data-select="multi" data-scope="single" data-typeahead="true">
            <div class="slds-form-element is-required">
                <label class="slds-form-element__label slds-text-body--regular" for="request_type">
                    {{labels.createRequest.requestType}} <abbr class="slds-required" title="required">*</abbr>
                </label>
                <div class="slds-form-element__control">
                    <!-- Trading partner radio box in request type -->
                    <span class="slds-radio slds-m-bottom--x-small" ng-if="!initialProductId">
                        <span ng-if="draftType === 'tp' || draftType === 'new'">
                            <input type="radio" name="request_type" ng-model="data.requestType" value="tp" id="
                            radio_request_type_tp" ng-change="clearUIOnSelectionChange()"/>
                            <label class="slds-radio__label" for="radio_request_type_tp">
                                <span class="slds-radio--faux"></span>
                                <span class="slds-form-element__label color_black" >{{labels.createRequest.tradingPartner}}</span
                                >
                            </label>
                        </span>

                        <div class="adjust_radio_input_text_field slds-m-bottom--xx-small slds-p-left--large slds-grid
                        float-label" ng-if="data.requestType === 'tp'">
                            <span ng-class="{'addTop': draftType === 'new'}" style="margin-right: 5px;">{{labels.common.to}} </
                            span>
                            <!-- Lookup for Trading Partner -->
                            <slds-lookup variable="sl.tradingPartner" model="data.selectedPartners" multiple="true" search-action
                            ="tradingPartnersConnectedWithMe" placeholder="labels.common.placeholderSelectTradingPartner"
```

Custom Labels referenced at HTML(View)

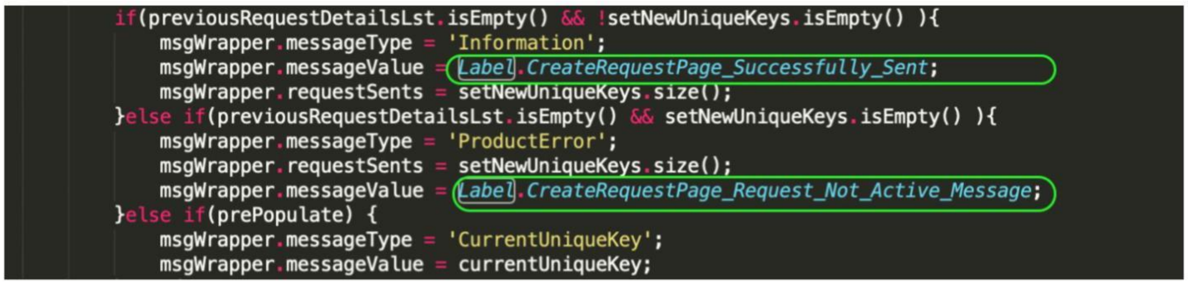


```javascript
        $scope.showCCAndBCCButton = false;
        $scope.showCCAndBCCInterface = false;
        $scope.showccBox = false;
        $scope.showBccBox = false;

        $scope.sendRequestDialog = {
            actions: [
                { text: $scope.labels.common.no,  close: true, click: $scope.sendExplicitNo },
                { text: $scope.labels.common.yes, close: true, click: $scope.sendExplicit, disable: function(row){ return $scope.disa
            ]
        };

        $scope.deleteDraftDialog = {
            actions: [{ text: $scope.labels.common.delete, close: true, click: $scope.cleanUpAndCancelForDraft }]
        };

        $scope.sendRequestTable = {
            columns: [
                { field:"isSelected",        label: " ", type: "checkbox", align: "left", change: $scope.checkBoxesChecked },
                { field: "productName",      label: $scope.labels.common.columnProductName, click: $scope.selectProduct},
                { field: "partnerName",      label: $scope.labels.common.columnPartnerName, click: $scope.selectPartner},
                { field: "formName",         label: $scope.labels.common.columnFormName},
                { field: "status",           label: $scope.labels.common.columnStatus, style: function(row){ if(row && row.status ==
                { field: "dueDate",          label: $scope.labels.common.columnDueDate, align: "center"},
                { field: "expirationDate",   label: $scope.labels.common.columnExpirationDate, align: "center"}
            ]
        };

        $scope.newRequestTable = {
            columns: [
                { field:"isSelected",        label: " ", type: "checkbox", align: "left", change: $scope.checkBoxesChecked },
                { field: "productName",      label: $scope.labels.common.columnProductName, click: $scope.selectProduct},
                { field: "partnerName",      label: $scope.labels.common.columnPartnerName, click: $scope.selectPartner},
                { field: "formName",         label: $scope.labels.common.columnFormName},
                { field: "dueDate",          label: $scope.labels.common.columnDueDate, align: "center"}
            ]
        };

        $scope.deleteDraftflag =  NavigationService.getParameter("recid") !== null;
        $scope.requestId = NavigationService.getParameter("recid");
        $scope.$watch('data.selectedDocumentTemplates', $scope.validateSelectedContainers);

         $scope.$watch('data.selectedProducts', function(newValue, oldValue) {
            if(Array.getCount($scope.data.selectedProducts) > 1) {
                for (var i = $scope.data.selectedDocumentTemplates.length; i--;) {
                    if ($scope.data.selectedDocumentTemplates[i].ParentDataName == 'Product Test')
```

Custom Labels referenced from AngularJs(controller)

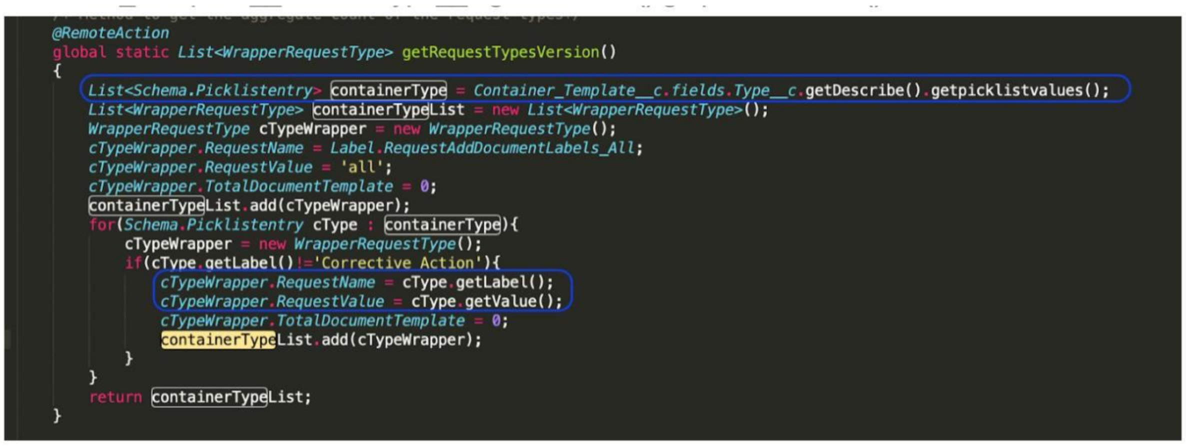
How to reference a Custom Label from Apex code

In order to reference custom labels from Apex code we use the Label class followed by the API name of the custom label **(Label.LabelApiName)** as shown in the following screenshot:



```
if(previousRequestDetailsLst.isEmpty() && !setNewUniqueKeys.isEmpty() ){
    msgWrapper.messageType = 'Information';
    msgWrapper.messageValue = Label.CreateRequestPage_Successfully_Sent;
    msgWrapper.requestSents = setNewUniqueKeys.size();
}else if(previousRequestDetailsLst.isEmpty() && setNewUniqueKeys.isEmpty() ){
    msgWrapper.messageType = 'ProductError';
    msgWrapper.requestSents = setNewUniqueKeys.size();
    msgWrapper.messageValue = Label.CreateRequestPage_Request_Not_Active_Message;
}else if(prePopulate) {
    msgWrapper.messageType = 'CurrentUniqueKey';
    msgWrapper.messageValue = currentUniqueKey;
```

In order to reference picklist translations we use **Schema.Picklistentry** class, doing a **"Describe"** over the object field we need to obtain:
Container_Template__c.fields.Type__c.getDescribe().getpicklistvalues().



```
@RemoteAction
global static List<WrapperRequestType> getRequestTypesVersion()
{
    List<Schema.Picklistentry> containerType = Container_Template__c.fields.Type__c.getDescribe().getpicklistvalues();
    List<WrapperRequestType> containerTypeList = new List<WrapperRequestType>();
    WrapperRequestType cTypeWrapper = new WrapperRequestType();
    cTypeWrapper.RequestName = Label.RequestAddDocumentLabels_All;
    cTypeWrapper.RequestValue = 'all';
    cTypeWrapper.TotalDocumentTemplate = 0;
    containerTypeList.add(cTypeWrapper);
    for(Schema.Picklistentry cType : containerType){
        cTypeWrapper = new WrapperRequestType();
        if(cType.getLabel()!='Corrective Action'){
            cTypeWrapper.RequestName = cType.getLabel();
            cTypeWrapper.RequestValue = cType.getValue();
            cTypeWrapper.TotalDocumentTemplate = 0;
            containerTypeList.add(cTypeWrapper);
        }
    }
    return containerTypeList;
}
```

NOTE: Since using **"Describe"** works as a query we never use this class and methods inside a loop.

When looping in order to obtain the results and retrieve them at the frontend we use a Wrapper class with the fields **"Label, Value"** to keep the relationship between them simple. To obtain the label use the method **getLabel()** and for the value use **getValue()** as shown in the previous screenshot.

Learn more from our team here and check out our services.