**What this project does??**

This project explains how Struts supports internationalization.

**Before starting this project, please go through theoretical part of this concept in 'Struts2 Notes' . That provide explanation to each section of this project.**

**How to create this project??**

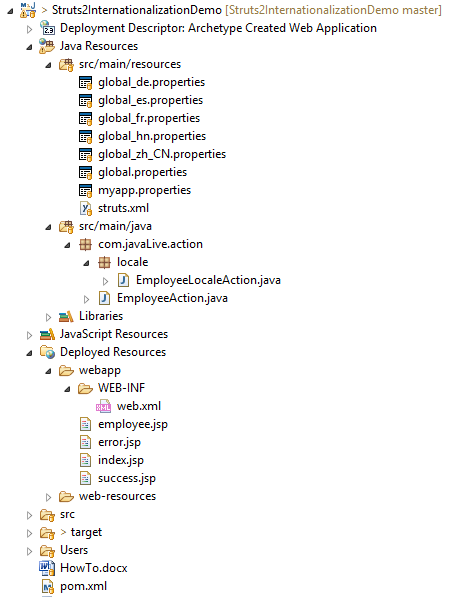
a) This is maven web project.

(Please refer 'CreateMavenWebProjectWithEclipse.docx' file in any other struts project where it is available to get idea about creating such project.)

b) Add require maven dependencies

c) Create and place packages in the respective locations as shown. **Pay special attention to main/src/resources/struts.xml (for mapping of actions).**

**Project structure is as follows:**

****

**Internationalization (i18n)** is the process of planning and implementing products and services so that they can easily be adapted to specific local languages and cultures, a process called localization. The internationalization process is sometimes called translation or localization enablement. Internationalization is abbreviated **i18n** because the word starts with an i and ends with an n, and there are 18 characters between the first i and the last n.

Will create a simple employee form screen, display the message from resource bundle via the Struts 2 UI components, and change the locale base on the selected language option.

**Resource Bundles:**

uses resource bundles to provide multiple language and locale options to the users of the web application. You don’t need to worry about writing pages in different languages. All you have to do is to create a resource bundle for each language that you want. The resource bundles will contain titles, messages, and other text in the language of your user. Resource bundles are the file that contains the key/value pairs for the default language of your application.

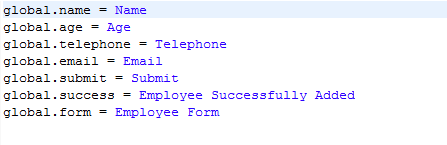
When you reference a message element by its key, Struts framework searches for a corresponding message bundle in the following order:

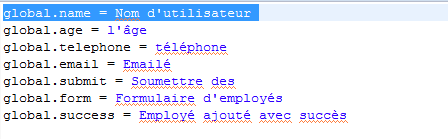
* ActionClass.properties
* Interface.properties
* SuperClass.properties
* model.properties
* package.properties
* struts.properties
* global.properties

To develop your application in multiple languages, you would have to maintain multiple property files corresponding to those languages/locale and define all the content in terms of key/value pairs.

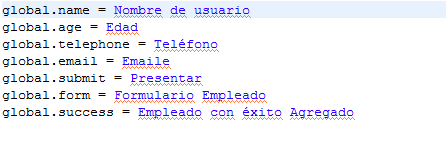
For example if you are going to develop your application for US English (Default), Hindi,Spanish, Chinese, German and Franch the you would have to create three properties files. Here I will use global.properties file only, you can make use of different property files to segregate different type of messages.

**global.properties:** By default English (United States) will be applied

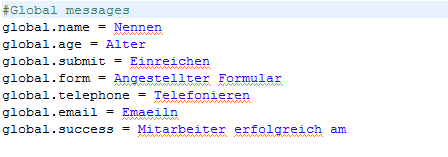
**global\_fr.properties:** This will be used for Franch locale.



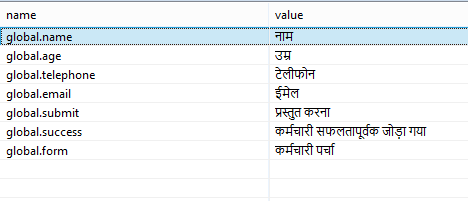
**global\_es.properties:** This will be used for Spanish locale.

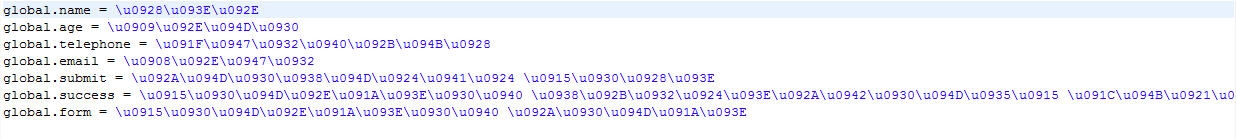


**global\_de.properties:** This will be used for German locale.

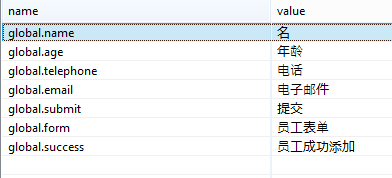


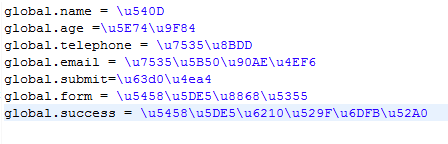
**global\_hn.properties:** This will be used for Hindi locale.





**global\_zh\_CN.properties:** This will be used for China locale.





**Access the messages:**

There are several ways to access the message resources, including ***getText***, the ***text tag, key***attribute of UI ***tags***, and the***i18n*** tag. Let us see them in brief:

To display i18n text, use a call to ***getText*** in the property tag, or any other tag, such as the UI tags as follows:

<s:property value="getText('some.key')" />

The text tag retrieves a message from the default resource bundle i.e.***struts.properties***

<s:text name="some.key" />

The ***i18n*** tag pushes an arbitrary resource bundle on to the value stack. Other tags within the scope of the ***i18n*** tag can display messages from that resource bundle:

<s:i18n name="some.package.bundle">

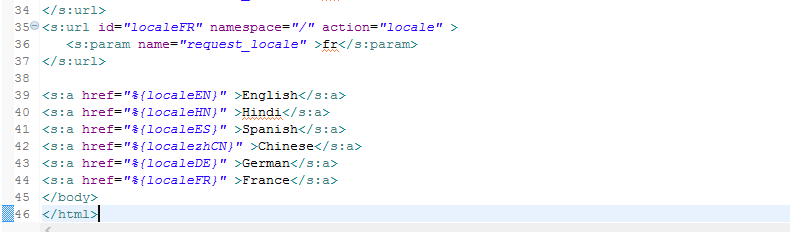
<s:text name="some.key" />

</s:i18n>

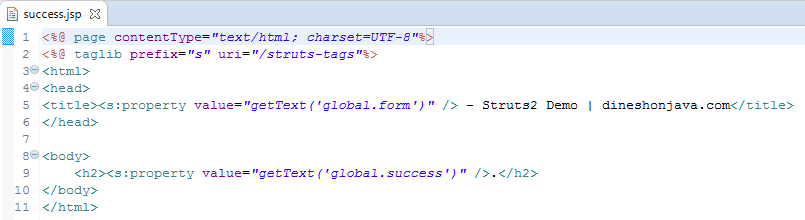
The key attribute of most UI tags can be used to retrieve a message from a resource bundle:

<s:textfield key="some.key" name="textfieldName"/>

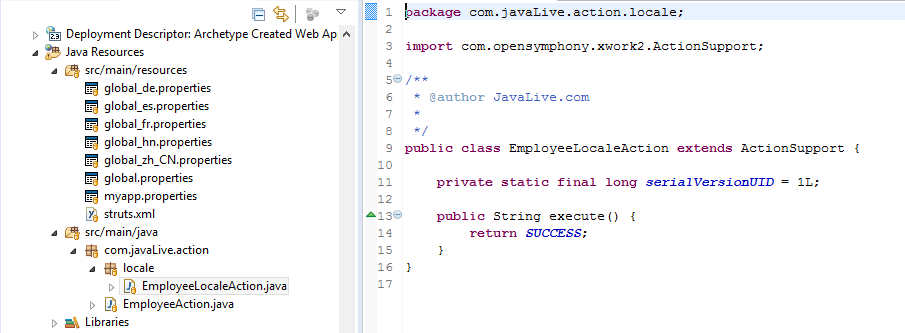
Let us target to create employee.jsp from the previous chapter in multiple languages. Same file would be written as follows:

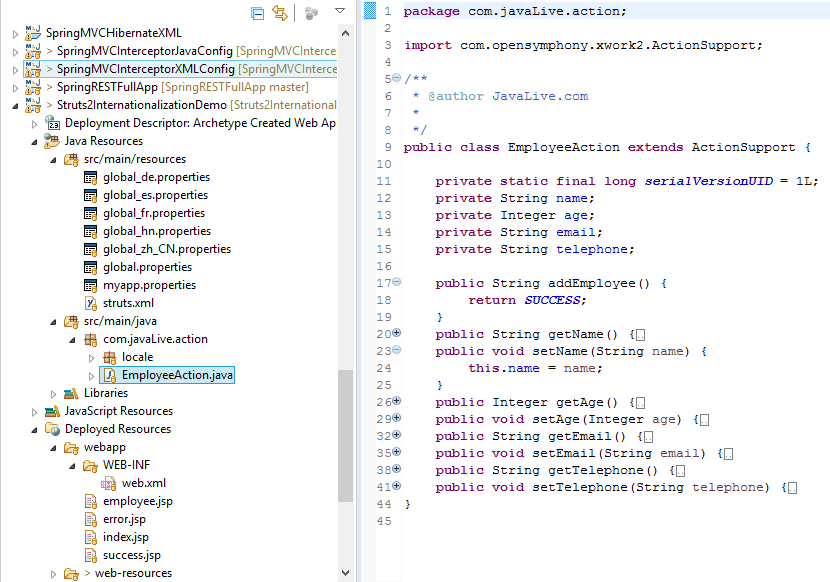
****

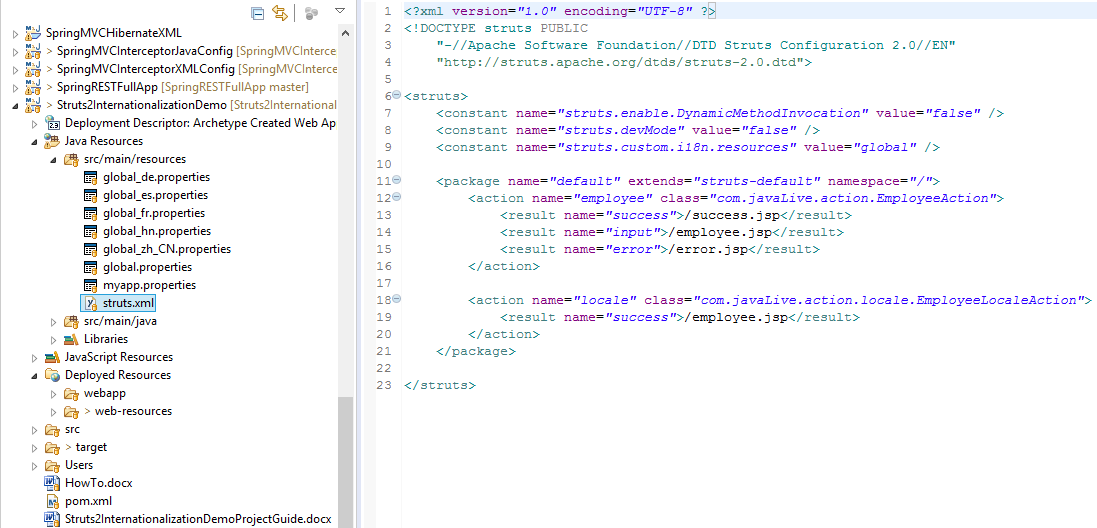
We will create ***success.jsp*** file which will be invoked in case defined action returns **SUCCESS**.

**Here we would need to create following two actions.**

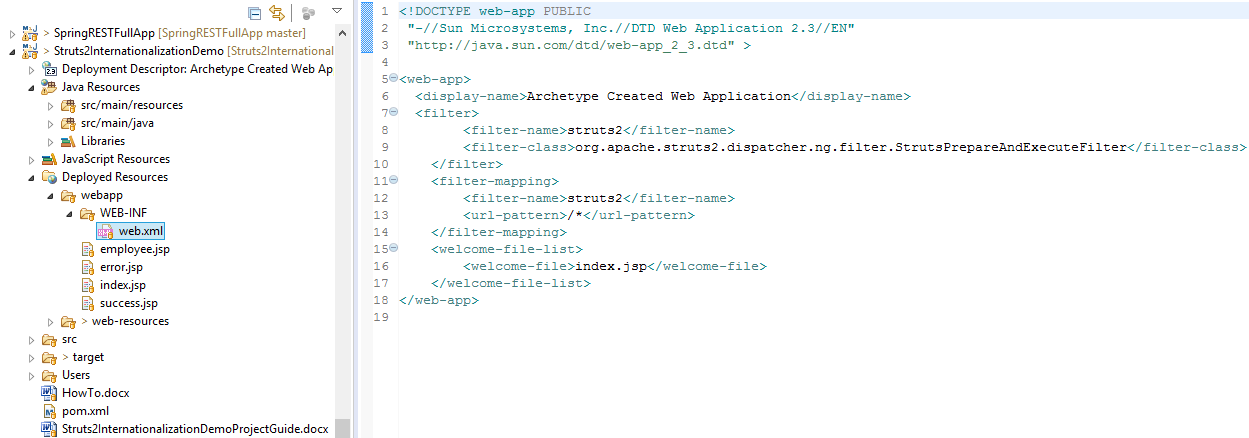
(a) First action a to take care of Locale and display same *success.jsp* file with different language

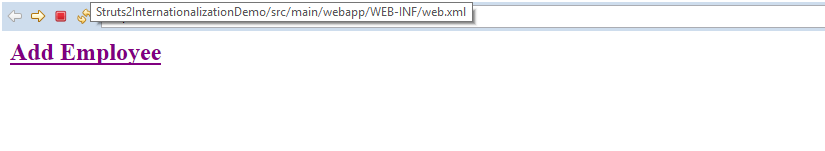
  
(b) Another action is to take care of submitting form itself.  
Both the actions will return SUCCESS, but we will take different actions based on return values because our purpose is different for both the actions:

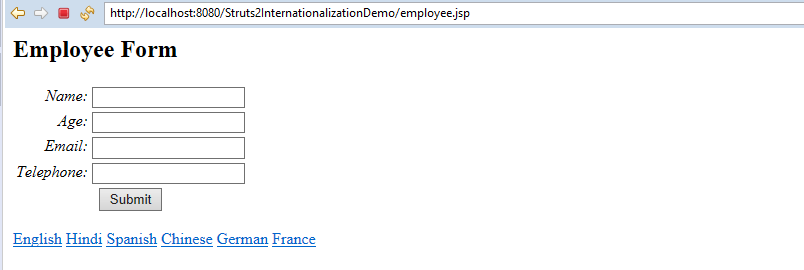
****

We will create our ***struts.xml*** with two actions as follows: ****

Following is the content of ***web.xml***  file:

****

****

****

Now click on Spanish, you will get following screen:



Same is the case for any other language. You will get form with respective language on clicking on link of particular language.

Here is how internationalization works in Struts2.