118n in spring boot MVC application

118n :: Internationlization (I 18 letters n)

=> Making our app working for different Locales is called enabling I18n on the application

=> Locale means language + country

eg:: en-US (english as it speaks in USA)

fr-FR (french as it speaks in France)

hi -IN (hindi as it speaks in India)

fr- CA (french as it speaks in Canada)

and etc..

=>By enabling 118n on the web application we can change the fllowing presentations according the Locale that is chosen

a) Presentation Lables

b) Date patterns

c) Time patterns

=>118n on the Project is no way related to Business Logic.. it all about changing the presentation logic as required for the different Locales

d) Number formats

e) Currency Symbols

f) Content indentation (most of languages left to right

and etc..

but urdu,arabic and etc.. right to left)

eg: google home page, gmail inbox page, youtube, facebook,amazon and etc..

=> By adding 118n support we can sell our software products to more clients .. if they are web applications then we can attract more customer belonging different coutries and localaties.

use

=> For presentation labels of I18n we need to multiple properties files for multiple locales on 1 per Locale basis

App.properties (base properties file -- english lables) App_fr_FR.properties (for french --> french labels) App_de_DE.properties (for german --> german labels) App_hi_IN.properties (for hindi --> hindi labels)

and etc..

The base file name of properties file

must reflect in other Propeties files like as show' here..

All these properties file must have same key but different values collected from google translator

home page

welcome to spring

register customer

french german hindi

english

customer_register.jsp

registation paga

customer name ::

customer addrs ::

customer bill Amount::

register

french german hindi

english

=>generally the content writers are responsible to write the content of the web pages in different locales ..nowadays the same work is simplified with the support of AI tools

step1) create spring starter project adding the following dependencies

a) lombok b) spring web c) jstl (apache jstl)

(collect from mvnrepository.com)

step2) prepare multiple properties for multiple locales as shown above having same keys with different values collected from google translator.

src/main/resources

static templates

application.properties myfile_de_DE.properties myfile_fr_FR.properties myfile_hi_IN.properties myfile.properties

myfile.properties

#Base file (english lables)

for 118n

=>any <filename>.properties can be taken as the

base properties file

=>This base properties file name must refect in Locale specific properties file as prefix name

=> if matching Locale specific properties is not found them it uses Base properties file to collect the content. myfile_hi_IN.properties

home.title=Welcome to Spring boot MVC home.link= register customer

cust.registration.title=Customer Registration Page cust.registration.name=Customer name cust.registration.addrs=Customer Address

cust.registration.billAmt-Customer BillAmount

cust.btn.register= register

#Hindi Hocale (hindi lables)

home.title= \u090f\u092e\u0935\u0940\u0938\u0940 \u092e\u0947\u0902 \u0906\u092a\u0915\u093E \u0938\u094d\u0935\u093e\u0917\u0924 \u0939\u0948 home.link= \u0917\u094d\u0930\u093e\u0939\u0915 \u092A\u0902\u091c\u0940\u0915\u0943\u0924 \u0915\u0930\u0947\u0902

cust.registration.title=\u0917\u094d\u0930\u093e\u0939\u0915 \u092a\u0902\u091c\u0940\u0915\u0930\u0923 \u092a\u0943\u0937\u094d\u0920 cust.registration.name=\u0917\u094d\u0930\u093e\u0939\u0915 \u0915\u093e \u0928\u093E\U092E cust.registration.addrs=\u0917\u094d\u0930\u093e\u0939\u0915 \u0915\u093E \u092a\u0924\u093E

cust.registration.billAmt=\u0917\u094d\u0930\u093e\u0939\u0915 \u092c\u093f\u0932

\u0930\u093e\u0936\u093F cust.btn.register=\u092a\u0902\u091c\u0940\u0915\u0943\u0924 \u0915\u0930\u0947\u0902

myfile_de_DE.properties

#german Locale

home.title=Willkommen bei Spring Boot mvc

home.link Kunde registrieren

myfile_fr_FR.properties

#French Locale (french lables)

home.title=Bienvenue sur Spring Boot MVC home.link= enregistrer le client

cust.registration.title=Page d'enregistrement client

cust.registration.name=Nom du client cust.registration.addrs-Adresse du client

cust.registration.billAmt-Rechnungsbetrag des Kunden

cust.btn.register=S'inscrire

cust.registration.title=Kundenregistrierungsseite cust.registration.name=Kundenname

cust.registration.addrs-Rechnungsbetrag des Kunden cust.registration.billAmt-Montant de la facture client cust.btn.register=registrieren

step3) Cfg base properties file in application.properties

In application.properties

#configure base properties file spring.messages.basename=myfile

=>keys in properties files are user-defined but all properties files must have same keys and values should be different collected from the Google translator..

step3) Activate 118n in spring boot MVC Application by configuring SessionLocaleResolver as the spring bean

In main class

@Bean(name="localeResolver") //fixed bean id

public SessionLocaleResolver createSLResolver() {

}

SessionLocaleResolver resolver=new Session LocaleResolver(); resolver.setDefaultLocale(new Locale("en","US"));

return resolver;

The moment this spring bean class obj is created.. it makes the underlying spring mvc or spring boot mvc app to activate the 118n on the application.

step4) Configure LocalChangeInterceptor as spring bean in main class using @Bean method

Resolvers are given to activate

certain facility in spring mvc

or spring boot mvc appliation

come

which will not automatically

eg: TilesResolver (To activate tile framework) SessionLocaleResolver (To activate 118n)

CosMultipartResolver (To activate file uploading) and etc...

This interceptor takes the locale value from specified request param and changes locale of every request by trapping the request.

```
@Bean
public LocaleChangeInterceptor createLCInterceptor() {
LocaleChangeInterceptor interceptor-new LocaleChangeInterceptor(); interceptor.setParamName("lang");
//default is locale
return interceptor;
}
```

=>This Interceptor

allows for changing the current locale on every request, via a configurable request parameter (default

parameter name: "locale").

step5) Develop Custom Configiurer class as spring bean to register the above interceptor with InterceptorRegistry

```
//MyWebMVCConfigurer.java
package com.nt.config;
import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.stereotype.Component;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry; import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer; import
org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
@Component
public class MyWebMVCConfigurer implements WebMvcConfigurer { @Autowired
private LocaleChangeInterceptor interceptor;
@Override
public void addInterceptors(InterceptorRegistry registry) {
registry.addInterceptor(interceptor);
}
}
```

The interceptors in spring boot mvc web application will be activated only after registering with InterceptorRegistry

step6) Develop the Model class

```
package com.nt.model;
import lombok.Data;
@Data
public class Customer {
private Integer cno;
private String cname;
private String caddrs;
```

}

private Float billAmount;

step7) develop the Controller class having handler methods to lanunch home pag, form page

//controller class

package com.nt.controller;

import java.util.Map;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import com.nt.model.Customer;

DispatcherServlet

(c) (g)

@Controller

(k) (n)

public class CustomerOperationsController {

(r) (v)

(i)

@GetMapping("/")

(e?) (t?)

(z) (c1)

public String showHome() {

(x)

//return LVN

for hompage

return "welcome"; (i) (y)

(g1) (k1) (01) (r1)

}

@GetMapping("/register")

(i1?)

(m1) public String showCustomerFormPage(@ModelAttribute("cust") Customer cust,

Map<String,Object> map) {

(d) (f) (s)

(u) (h1) (j1)

HandlerMapping

Locale

(h) --> activates default from base properties file (myfile.properties) LocaleChangeInterceptor (w) --> since lang=hi_IN activates hindi Locale from myfille_hi_IN.properties file since lang=hi_IN ,so activates hindi locale (myfile_hi_IN.properties)

(11)-->

InternalResourceViewResolver

|-->prefix: /WEB-INF/pages/ |--->suffix:.jsp (1) (a1) (p1) View obj

(m) (b1)

WEB-INF/pages/welcome.jsp WEB-INF/pages/customer_register.jsp

(91)

For formpage

launching

}

}

//return LVN

return "customer_register"; (n1)

step8) Develop jsp pages enabling UTF-8 content type and reading locale specific messges from properties files.

WEB-INF/pages/welcome.jsp

(0) (d1)

<%@page isELIgnored="false" contentType="text/html; charset=UTF-8" %> <%@taglib uri="http://www.springframework.org/tags" prefix="sp"%>

<h1 style="color:blue;text-align:center"> <sp:message code="home.title"/></h1>

<br><br> (f1)

(for <sp:message> tag)

(P) collects

<a href="register"><h2 style="color:red;text-align:center"><sp:message code="home.link"/></h2> </a> messages from

<br><br>

(e1) collects the messages

from myfile_hi_IN.properties file

default properties

file (myfile.properties)

(english labels)

```
<p align="center">
<a href="?lang=fr_FR">French</a> &nbsp;&nbsp;&nbsp;
<a href="?lang=de_DE">German</a> &nbsp;&nbsp;&nbsp;
<a href="?lang=hi_IN">
(q)</a> &nbsp;&nbsp;&nbsp;
<a href="?lang=en_US">English</a>
</p>
```

=>if <a> tag href not having url or href itself not placed

then the generated request goes to that url using

which the web page is launched.

WEB-INF/pages/customer_register.jsp (s1)

```
<%@ page isELIgnored="false" contentType="text/html; charset=UTF-8" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@taglib uri="http://www.springframework.org/tags" prefix="sp"%>
<h1 style="color:red;text-align:center"><sp:message code="cust.registration.title"/></h1>
<form:form modelAttribute="cust">
<table border="1" align="center" bgcolor="cyan">
<tr>
<td><sp:message code="cust.registration.name"/> </td>
<td><form:input path="cname"/> </td>
(t1)
</tr>
collects the message from
<tr>
myfile_hi_IN.properties file
<td><sp:message code="cust.registration.addrs"/> </td>
<td><form:input path="caddrs"/> </td>
</tr>
<tr>
<td><sp:message code="cust.registration.billAmt"/> </td>
<td><form:input path="billAmount"/> </td>
</tr>
<tr>
<td><input type="submit" value="<sp:message code="cust.btn.register"/>"/> </td>
</tr>
</table>
```

</form:form>

<br><br>

<p align="center">

<a href="?lang=fr_FR">French</a> &nbsp;&nbsp;&nbsp;

<a href="?lang=de_DE">German</a> &nbsp;&nbsp;&nbsp;

<a href="?lang=hi_IN"> f

</a> &nbsp;&nbsp;&nbsp;

<a href="?lang=en_US">English</a>

</p>

(a)During the deployment of web application all singleton scope spring beans like SessionLocaleResolver, LocaleChangeInterceptor, CustomerOperationsController, MyWebMVCConfigurer classes will be pre-instantiated and the the injections takes place

=>*SessionLocaleResolver obj creation activates the 118n*

=>*LocaleChnageInterceptor will be activated becoz it is registered with InterceptorRegistry*

(b) http://localhost:2020/BootMVCProj15-118n

Applying Date,time, number,currency formts in the 118n App

=>We can use JSTL core, formatting tag libraries together to format date, time, number

and currency values as show below.

step1) add JSTL libaries to build path

<!-- https://mvnrepository.com/artifact/org.eclipse.jetty/apache-jstl -->

<dependency>

<groupId>org.eclipse.jetty</groupId>

<artifactId>apache-jstl</artifactId>

<version>11.0.0</version>

=>JSTL is combination of 5 jsp taglibaries .. all these taglibrary tags will be

used make jsp pages as the java code less jsp pages..

a) core b) sql c) formatting d) functions e) xml

</dependency>

step2) Import JSTL core, formatting tag libaries

welcome.jsp

<%@page isELIgnored="false" contentType="text/html; charset=UTF-8" %> <%@taglib uri="http://www.springframework.org/tags" prefix="sp"%> <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %> <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<h1 style="color:blue;text-align:center"> <sp:message code="home.title"/></h1>

<br><br>

=> In every jsp page, there are 9 implicit objs

i)page ii)pageContext iii)request iv) response v)session

vi) application vii) exception viii)out ix) config

=> Implicit ref variables in java app are "this","super"

=> implicit threads in java app are "main", "gc"

=> implicit streams in java app are "System.in","System.out", "System.err"

=> implicit properties in java app are :: "length","class"

```
<a href="register"><h2 style="color:red;text-align:center"><sp:message code="home.link"/></h2> </a>

←

<br>

<h1>Current active Locale is :: ${pageContext.response.locale} </h1>

<fmt:setLocale value="${pageContext.response.locale}"/>

<jsp:useBean_id="dt" class="java.util.Date"/>

___<fmt:formatDate var="fdt" value="${dt}" type="date" dateStyle="SHORT" />

<b>formatted date :: ${fdt} </b><br>

<fmt:formatDate var="ftime" value="${dt}" type="time" timeStyle="FULL" />

<b>formatted time :: ${ftime} </b>

<fmt:formatNumber var="fnumber" value="1000000" type="number"/><br>

<b>formatted number :: ${fnumber}</b>

<fmt:formatNumber var="fcurrency" value="1000000" type="currency"/><br>

<b>formatted currency :: ${fcurrency}</b>

<fmt:formatNumber var="fpercentage" value="0.211" type="PERCENT"/><br> <b>formatted percentage :: ${fpercentage}</b>

<br><br>

<p align="center">

<a href="?lang=fr_FR">French</a> &nbsp;&nbsp;&nbsp;

<a href="?lang=de_DE">German</a> &nbsp;&nbsp;&nbsp;

<a href="?lang=hi_IN">

हिन्दी </a> &nbsp;&nbsp;&nbsp;

<a href="?lang=en_US">English</a>

</p>
```

localhost:2525/MVCBootProj14-118n/?lang=hi_IN

```
current active Locale is :: hi_IN

Formatted date :: शुक्रवार, 28 अक्तूबर 2022

Formatted time :: 8:20:04 am HRÂY

Formatted Number:: 100,000,000

Formatted currency :: 100,000,000.00

Formatted percent :: 75%

French

HI-6 GHU
```

English

Germany हिन्दी

chinese

M5 MVCBootProj14-118n [boot]

> Deployment Descriptor: MVCBootProj14-118n

>Spring Elements

> JAX-WS Web Services

src/main/java

com.nt

>

> MvcBootProj14118nApplication.java

>ServletInitializer.java

com.nt.configurer

> MyWebConfigurer.java

com.nt.controller

> CustomerOperationsController.java

com.nt.model

> Customer.java

#src/main/resources

static

templates

application.properties

myfile_de_DE.properties myfile_fr_FR.properties myfile_hi_IN.properties

myfile_zh_CN.properties

myfile.properties

src/test/java

> JRE System Library [JavaSE-17]

> Maven Dependencies

> Deployed Resources

✓

src

✓

main

> java

> resources

webapp

✓ WEB-INF

✓ pages

customer_register.jsp

>test

> target

WHELP.md

mvnw

mvnw.cmd

Mpom.xml

welcome.jsp

स्प्रिंग बूट **MVC** में आपका स्वागत है **- 118n**

ग्राहक पंजीकृत करें