Labo Spring 5
- *Internationalisation* -

Objectifs

- Traduire les libellés statiques et les messages d'erreur en plusieurs langues
- Afficher les libellés statiques et messages d'erreur dans la langue de la locale
- Permettre à l'utilisateur de choisir sa langue et afficher les libellés statiques et messages d'erreur dans la langue choisie par l'utilisateur

Étape 1 : Création des fichiers de traduction des libellés statiques

Fichier de libellés statiques par langue (de type *property*)

Un fichier de type dictionnaire est créé pour chaque langue ; la clé de chaque entrée du dictionnaire est identique pour toutes les langues et la valeur varie d'une langue à l'autre.

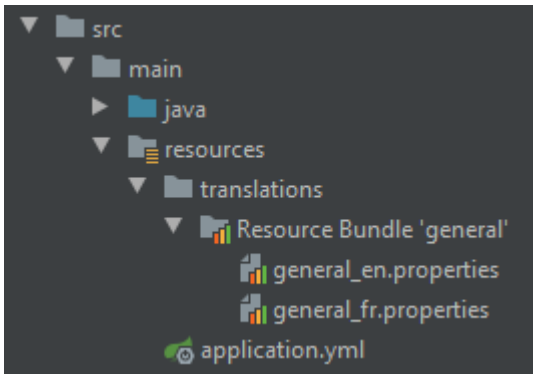
Le nom de chaque fichier est composé d'une même racine suivie de "_" suivi du code de la langue (2 lettres).

Spring se charge de récupérer la valeur de chaque libellé statique sur base de sa clé et de la locale courante.

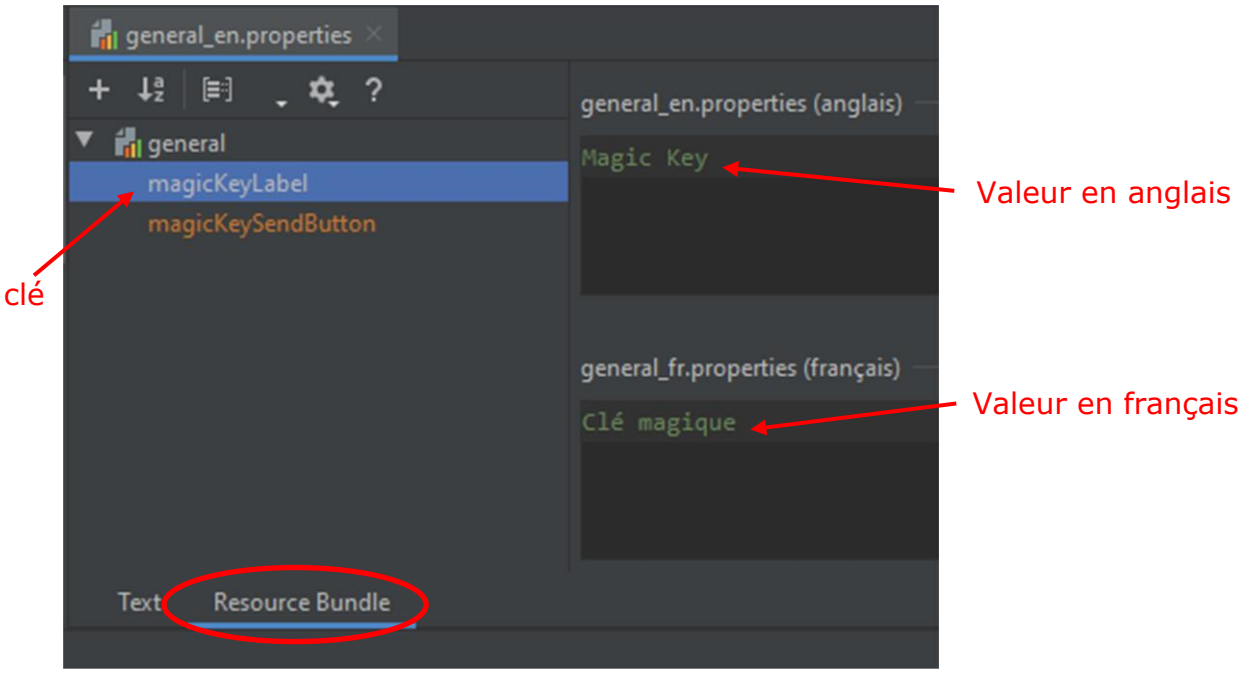
Créez un répertoire *translations* dans le répertoire *resources* situé dans *src/main*.

Créez ensuite dans ce répertoire *translations* un Resource Bundle intitulé "*general*" qui contiendra les différents fichiers de traduction pour les libellés statiques de l'application Web :

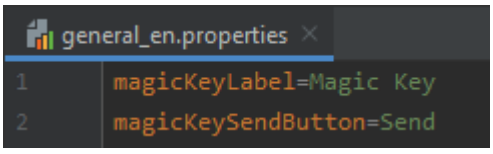
Clic droit sur *translations* ⇨ new ⇨ Resource bundle ⇨ base name : *general* ⇨ Locales to add + ⇨ Input locale code to add : en ajouter au moins deux (*en* et *fr*)



Remplissez les fichiers de traduction en anglais et en français : clic droit sur un fichier de propriétés (ex : *general_en.properties*) ⇨ basculez de l'onglet *Text* vers l'onglet *Resource Bundle* et remplissez les fichiers de traduction en choisissant une clé et en donnant les valeurs dans chaque langue correspondant à cette clé.



Contenu d'un fichier de traduction (onglet *Text*) :



Étape 2 : Mise à jour de la classe *MainConfiguration*

Ajoutez deux déclarations de beans dans la classe *MainConfiguration*. Ceux-ci se chargeront de la traduction des libellés statiques de l'application en fonction de la locale et de la localisation des fichiers de traduction.

Vous pouvez vous inspirer du fichier mis à votre disposition sur Moodle.

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ResourceBundleMessageSource;
import org.springframework.validation.DefaultMessageCodesResolver;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MainConfiguration implements WebMvcConfigurer {

    @Bean
    public DefaultMessageCodesResolver defaultMessageCodesResolver() {
        DefaultMessageCodesResolver defaultMessageCodesResolver = new DefaultMessageCodesResolver();
        return defaultMessageCodesResolver;
    }

    @Bean
    public ResourceBundleMessageSource messageSource() {
        ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
        messageSource.setDefaultEncoding("UTF-8");
        messageSource.setBasenames("translations/general","translations/errors");
        messageSource.setUseCodeAsDefaultMessage(true);
        return messageSource;
    }
```

Répertoire des fichiers de traduction

Racine des noms de fichiers

Traduction des messages d'erreur

Étape 3 : Adaptation des pages Web

Dans les pages jsp, adaptez le code afin que les fichiers de traduction soit pris en compte. Chaque libellé statique doit être remplacé par l'appel à la balise *<spring :message …>* en utilisant le paramètre *code* pour préciser la clé des fichiers de traduction. Par exemple, dans la page *welcome.jsp*, l'ancien label "*Magic Key*" doit désormais être traduit dans la langue de la locale :

```
<form:label path="magicKey">
    <spring:message code="magicKeyLabel" />
</form:label>
```

Faites-en sorte également que le libellé du bouton *Send* soit traduit dans la langue de la locale.

Testez votre application. En fonction de la locale par défaut de votre ordinateur, le fichier de traduction correspondant sera utilisé.

Si vous avez des **problèmes d'accents** ou de caractères spéciaux qui ne s'affichent pas correctement, vérifiez les deux points suivants :

1. Dans la page *template* ainsi que dans la page *welcome.jsp*, vous devez avoir placé l'instruction qui précise que vous travaillez en UTF-8 :

template.jsp

```
1  <%@ page pageEncoding="UTF-8"
2           contentType="text/html; charset=UTF-8"%>
```

2. Les fichiers de traduction doivent être encodés en UTF-8. Pour vérifier ou modifier le type d'encodage d'un fichier texte, ouvrez ce fichier via Notepad++ et via l'option de menu "Encodage", choisissez le type UTF-8. Sauvez le fichier. Arrêtez votre application et redéployez-la. Si nécessaire, fermez le navigateur et retestez.

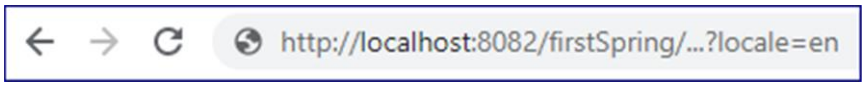
Étape 4 : Affichage des boutons proposant les langues

Choix de la langue par l'utilisateur

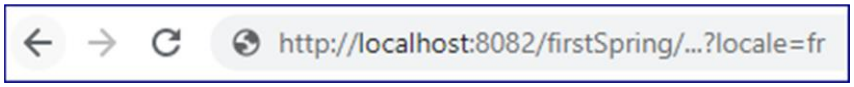
Le principe consiste à permettre à l'utilisateur de choisir sa langue sur le site Web. Un bouton par langue sera affiché. Un clic sur un de ces boutons doit avoir pour effet de changer la langue (locale) ; un paramètre sera ajouté à l'URL de la requête pour prévenir le serveur de la demande de changement de langue. Côté serveur, ce paramètre dans l'URL doit être intercepté. Cette nouvelle langue (locale) courante ainsi interceptée sera stockée sous forme d'un cookie de type session. C'est la valeur de la locale stockée dans le cookie que Spring utilisera pour déterminer les fichiers de traduction à choisir aussi bien pour les libellés statiques que pour les messages d'erreur.

Deux drapeaux vont être ajoutés au template, un représentant la langue anglaise et l'autre la langue française.

Un clic sur le drapeau anglais aura pour effet d'ajouter "*?locale=en*" à l'URL de la requête :



Un clic sur le drapeau français aura pour effet d'ajouter "*?locale=fr*" à l'URL de la requête.

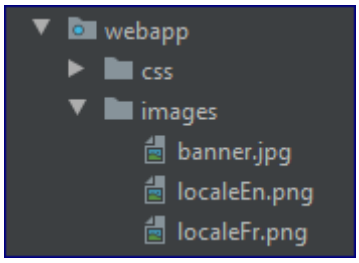


Pour ce faire, dans la partie *<head>* de la page *template.jsp*, déclarez deux variables de type URL appelées *localeEn* et *localeFr* en utilisant la balise *<spring :url>* ; chacune de ces url contiendra l'URL courant (value=""). Ajoutez à chacune de ces URL le paramètre intitulé *locale* ainsi que sa valeur, via la balise *<spring :param>*.

```
<spring:url var="localeFr" value="">
    <spring:param name="locale" value="fr" />
</spring:url>

<spring:url var="localeEn" value="">
    <spring:param name="locale" value="en" />
</spring:url>
```

Ajoutez les deux fichiers images correspondant aux drapeaux dans le répertoire images.



Ajoutez les deux liens vers ces images dans le template en veillant à utiliser dans *<a href=… />* les URL spring définies ci-dessus.

```
<a href="${localeFr}">
    <img alt="fr" src='<spring:url value="/images/localeFr.png"/>' />
</a>
```

Étape 5 : Mise à jour de la classe *MainConfiguration*

Mettez à jour la classe *MainConfiguration*.

Ecrivez le code permettant d'intercepter le paramètre appelé *locale* dans l'URL des requêtes.

Déclarez ensuite un bean qui se chargera de stocker la locale ainsi détectée dans un cookie de type session. La locale courante sera celle ainsi stockée dans ce cookie.

Vous pouvez vous inspirer du fichier mis à votre disposition sur Moodle.

```
import org.springframework.web.servlet.i18n.CookieLocaleResolver;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
```

```
@Override
public void addInterceptors(InterceptorRegistry registry) {
    LocaleChangeInterceptor interceptor = new LocaleChangeInterceptor();
    interceptor.setParamName("locale");
    registry.addInterceptor(interceptor);
}

@Bean
public LocaleResolver localeResolver() {
    CookieLocaleResolver resolver = new CookieLocaleResolver();
    resolver.setDefaultLocale(new Locale( language: "fr"));
    resolver.setCookieName("localeCookie");
    resolver.setCookieMaxAge(-1);
    return resolver;
}
```

Nom du paramètre à intercepter dans l'URL

Locale par défaut

De type session

Testez votre application.

Étape 6 : Traduction des messages d'erreur

Messages d'erreur en différentes langues

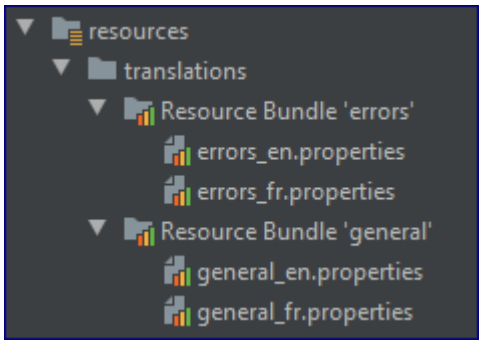
Les messages d'erreur gérés par Spring peuvent aussi être personnalisés et affichés en différentes langues. Un fichier de type dictionnaire est créé pour chaque langue.

La clé de l'erreur peut être constituée sur base des annotations utilisées dans les classes de type modèle pour la validation des formulaires.

Un message d'erreur peut être général, par exemple un même message d'erreur peut être prévu pour la violation du caractère obligatoire de toute variable de type *String* ; la clé de l'erreur dans le fichier de traduction est alors *NotNull.java.lang.String*.

Un message d'erreur peut être plus spécifique et porter sur un champ particulier de formulaire, par exemple en cas de violation de la valeur minimale acceptée pour le champ code postal (variable d'instance *zipCode* de la classe modèle correspondante) dans le formulaire *userForm*, la clé de l'erreur est *Min.userForm.zipCode*.

Créez dans le répertoire *translations* un Resource Bundle intitulé "*errors*" qui contiendra les différents fichiers de traduction des messages d'erreur de l'application Web (au moins en anglais et en français).



Personnalisez les messages d'erreur en anglais et en français liées à la validation du formulaire de création de l'utilisateur, en vous basant sur les annotations utilisées dans la classe *User* de la couche modèle.

Testez votre application en tentant de violer les contraintes liées au formulaire de création d'un nouvel utilisateur.