Internationalization

Summary

The E-government standard framework uses LocaleResolver provided by Spring MVC.
We'll examine LocaleResolver, application setting and how to get and utilize message resourcesapplied with multiple languages.

Spring MVC provides the type of LocaleResolveras shown below to support multiple languages.

- CookieLocaleResolver: use locale information using cookie
- SessionLocaleResolver: use locale information using session
- AcceptHeaderLocaleResolver: use locale information set in the browser of client

If it is not defined in Bean configuration file, AcceptHeaderLocaleResolver is applied by default.

Description

3 LocaleResolvers

CookieLocaleResolver

If setting CookieLocaleResolver, read the Locale set in the cookie of user.
samlple-servlet.xml

```
...
<bean id="localeResolver"
class="org.springframework.web.servlet.i18n.CookieLocaleResolver" >
<property name="cookieName" value="clientlanguage"/>
<property name="cookieMaxAge" value="100000"/>
<property name="cookiePath" value="web/cookie"/>
</bean>
...
```

Following properties can be used.

Property	Default Value	Description
cookieName	classname + locale	Cookie Name
cookieAge	nteger.MAX_INT	Set it to -1 and it will disappear when you close the browser
cookiepath	/	Designate the Path and it will refer in the relevant Path and the Path under it.

SessionLocaleResolver

Get locale information from session that the reqeust has.
samlple-servlet.xml

```
...
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver" />
...
```

AcceptHeaderLocaleResolver

Gets the Locale in the accept-language area of request header sent by user browser. Represent the locale of user browser.
samlple-servlet.xml

```
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver"
/>
```

XML Setting

Web Configuration

Apply Charset UTF-8 to the request incoming through web.

Set as follows to encode using CharacterEncodingFilter.

web.xml

```
...
<filter>
<filter-name>encoding-filter</filter-name>
<filter-class>
org.springframework.web.filter.CharacterEncodingFilter
</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>

<filter-mapping>
<filter-name>encoding-filter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
...
```

Spring Configuration

To implement in a way that allows the user to select and to directly select the language rather than using locale information of browser of user, use CookieLocaleResolver or SessionLocaleResolver. Since multiple languages should be supported first, the message have to be implemented after extracted to MessageSource.
Refer to Resourcefor more details on MessageSource
ThemessageSource was set as shown below.

samlple-servlet.xml

```
<bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">
<property name="basenames">
<list>
<value>classpath:/message/message</value>
</list>
</property>
</bean>
```

message properties files are as follows:
It is separated into ko, en depending on locale.

message_ko.properties

view.category=category

message_en.properties

view.category=category

ResourceBundleMessageSourceis the name ofbeannamesand receives messages. By default, it receives a message from messages.properties. It receives from messages_ko_Krproperties, if locale is Korean, and from messages_en_US.propertiesif it is English. Register localeResoverandlocaleChangeInterceptoras shown below, and register it in **DefaultAnnotationHandlerMapping**as an interceptor so that it can operate on the Annotation basis.

samlple-servlet.xml

```
<!—If using Locale with session-->
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver"/>

<!—If using Locale with cookie
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.CookieLocaleResolver"/>
-->
<bean id="localeChangeInterceptor"
class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
        <property name="paramName" value="language"/>
</bean>

<bean id="annotationMapper"
class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping">
        <property name="interceptors">
                <list>
                        <ref bean="localeChangeInterceptor"/>
                </list>
        </property>
</bean>
```

If set as above using SessionLocaleResolver, Locale determination is a **language** and is transferred to Request Parameter. You can see that category term is changed to English and Korean as shown below:

The screen showing the list, for example, is expressed as following, using the Springmessage tag:

```
<spring:message code="view.category" />



<%@ taglib prefix="spring" uri=http://www.springframework.org/tags %>

<form:formcommandName="message" >
....
                <table border="1" cellspacing="0" class="boardList" summary="List of Category">
                        <thead>
                                <tr>
                                        <th scope="col">No.</th>
                                        <th scope="col">
                                                <input name="checkAll" type="checkbox"
class="inputCheck" title="Check All" onclick="javascript:fncCheckAll();"/>
                                        </th>
                                        <th scope="col"><spring:message code="view.category"
/> ID</th>
                                        <th scope="col"><spring:message code="view.category"
/>Name</th>
                                        <th scope="col">Use or not</th>
                                        <th scope="col">Description</th>
```

```
                                        <th scope="col">Registrant</th>
                                    </tr>
                                </thead>
....
```

If executing the relevant page on the screen, it is as follows:

If Korean :

http://localhost:8080/sample-web/sale/listCategory.do?language=ko

If English :

http://localhost:8080/sample-web/sale/listCategory.do?language=en

Bring Local Application Message in Java Source

For your reference, MessageSourceis formed with the following methods.(In fact, the implement here is ResourceBundleMessageSource.)

Stringmsg = messageSource.getMessage(messageKey, messageParameters, defaultMessage, locale);

N. Reference