Spring使用拦截器支持国际化很方便，使用时只需要两个步骤：

## 一.spring配置

具体配置方式如下：

<!-- 资源文件绑定器，文件名称：messages.properties（没有找到时的默认文件）， messages\_en.properties（英文），messages\_zh\_CN.properties（中午），等等-->          
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">

<property name="basename" value="config.messages.messages" />

</bean>

<!-- 定义本地化变更拦截器 -->  
 <bean id="localeChangeInterceptor"  
  class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />

<util:list id="interceptors">  
<ref bean="localeChangeInterceptor" />  
</util:list>

<!-- 定义注解URL映射处理器 ，所有的请求映射必须关联本地化拦截器-->  
 <bean id="urlMapping"  
  class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping">  
  <property name="interceptors" ref="interceptors" />  
  <property name="order" value="1"></property>  
 </bean>

这时还需要本地化处理器进行处理，有三种处理器，如下：

<!-- 定义本地化处理器 -->

### 1. 基于session

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

### 2. 基于请求

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

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

以上三种处理器理论上配置任意一个就可以，不过有两点需要注意：

1）第二个不能直接使用，需要通过继承重写相应的方法，如下：

public class MyLocaleResolver extends AcceptHeaderLocaleResolver{

 private Locale myLocal;   
 public Locale resolveLocale(HttpServletRequest request) {   
   return myLocal==null?request.getLocale():myLocal;   
 }   
 public void setLocale(HttpServletRequest request, HttpServletResponse response, Locale locale) {   
   myLocal = locale;   
 }

}

修改配置文件中的localeResolver配置，class指向这个类就可以了。

2）第一与第三个用法相同，只不过前者使用session，session过期就需要重新设置，而后者使用cookie，可以根据项目的具体情况进行选择。

一般来说，使用第一种和第三种比较常见。

## 二.使用方法

当不做处理时默认会使用浏览器自己的语言设置，如果想改变语言，只需要在请求后面加上一个参数即可，

默认的参数名为locale，里面放的就是你的提交参数，如:en\_US，zh\_CN之类的，

所以，只需要在页面上加上相应的链接即可，如下：

<a href="xxx.do?locale=zh\_CN">中文</a>

<a href="xxx.do?locale=en">英文</a>

页面中使用jstl或spring标签，如下：

<fmt:message key="test.app"/>  
<s:message code="test.app"/>

无论使用哪种处理器，locale设置过一次就可以了，不需要每个连接后面都加上locale参数。

### 一、基于浏览器语言的国际化配置

使用Spring的MVC，并且配置中有配置Resource文件

Xml代码  [收藏代码](javascript:void())

1. <!-- 资源文件绑定器 -->
2. **<bean** id="messageSource"class="org.springframework.context.support.ResourceBundleMessageSource"**>**
3. **<property** name="basename" value="message-info" **/>**
4. **<property** name="useCodeAsDefaultMessage" value="true" **/>**
5. **</bean>**

其中，message-info是你的properties文件的通用名。如：我的配置文件叫message-info.properties,message-info\_zh\_CN.properties等等,只要有了这个配置，然后配置JSP渲染器为JSTL支持的，那么在你的JSP文件中使用fmt标记就可以实现客户浏览器语言国际化了。   
如：<fmt:message key="info.login.title" />   
其中的info.login.title和你的资源文件对应.   
另外一种方式是使用spring自带的标签显示国际化信息，如：   
<spring:message code="main.title" /><br>   
<input type="button" value="<spring:message code="main.title" />"/><br> 

### 二、基于动态加载的国际化配置

#### 1、基于请求的国际化配置

基于请求的国际化配置是指，在当前请求内，国际化配置生效，否则自动以浏览器为主。   
配置方式如下：   
首先配置拦截器   
<!-- 国际化操作 拦截器 必需配置，可以和其它国际化方式通用 -->        
<bean id="localeChangeInterceptor" class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />  
这个配置呢，是不论请求级别的国际化，还是Cookie级别的国际化，再或者Session级别的国际化，都必需有配置这个拦截器，否则会不能使用。   
配好上面的拦截器之后，就将拦截器注入到你的UrlHandlerMapping中，例如:

Xml代码  [收藏代码](javascript:void())

1. **<bean** id="defaultUrlMapping" class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"**>**
2. **<property** name="interceptors" ref="localeChangeInterceptor" **/>**
3. **<property** name="order"**>**
4. **<value>**1**</value>**
5. **</property>**
6. **</bean>**

这个时候，但凡有了符合UrlMapping的请求，就会被拦截，并且开始配置国际化参数   
        <bean id="localeResolver" class="org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver">        </bean>   
默认的参数名为locale主意大小写。里面放的就是你的提交参数。如:en\_US，zh\_CN之类的，这个时候，你在页面上加一句<a href="?locale=zh\_CN">简体中文</a>   
如果你的资源中，饱含建议中文的配置，那么就会变成你确定的简体中文拉。 

#### 2、基于Session的国际化配置

拦截器和基于请求的相同   
Session的配置如下：   
        <bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver"></bean>    
在你的处理的Controller中，将提交上来的locale字段信息生成真正的Locale对象，然后将对象保存在Session中，默认保存的ID是SessionLocaleResolver.LOCALE\_SESSION\_ATTRIBUTE\_NAME   
这样，当你的Session不过期，那么语言种类始终保持正确的说。我一直是这样子用的，我觉得还是Session的好，老外们用了很满意。 

#### 3、基于Cookie的国际化配置

这个我就不说了，反正用的不多，至少我做的项目能不用Cookie就不用Cookie，所以，基于Cookie的国际化配置我就不细说了，如果想知道怎么配置，那么下载一个Spring，其中的例子程序就是用Cookie配置的，你自己读代码就OK了。   
三、注意事项   
如果不用默认的浏览器语言国际化方式，那么拦截器一定要配置，如果你有多个UrlMapping，那么就每个都配上拦截器。   
至于配置的LocaleResolver的名字，一定要用上面的配置中的名字localeResolver当然了，这个是默认的名字来的，自己设置成别的也可以，但是就是麻烦，反正我用默认的就感觉不错   
解决问题:   
在前几天引用“Spring 的MVC I18N－国际化相关配置 ”并做了测试，发现 有一问题。程序运行会抛出异常   
“Cannot change HTTP accept header - use a different locale resolution strategy”，根本原因是spring source 做了限制,源码如下

Java代码  [收藏代码](javascript:void())

1. **public** **class** AcceptHeaderLocaleResolver **implements** LocaleResolver {
2. **public** Locale resolveLocale(HttpServletRequest request) {
3. **return** request.getLocale();
4. }
5. **public** **void** setLocale(HttpServletRequest request, HttpServletResponse response, Locale locale) {
6. **throw** **new** UnsupportedOperationException(
7. "Cannot change HTTP accept header - use a different locale resolution strategy");
8. }
9. }

请注意上面的类，该类允许继承，所以需要改写setLocale方法，源码示范如下

Java代码  [收藏代码](javascript:void())

1. **package** org.springframework.web.servlet.i18n;
2. **import** java.util.Locale;
3. **import** javax.servlet.http.HttpServletRequest;
4. **import** javax.servlet.http.HttpServletResponse;
5. **import** org.springframework.web.servlet.DispatcherServlet;
6. **import** org.springframework.web.servlet.LocaleResolver;
7. **public** **class** MyAcceptHeaderLocaleResolver **extends** AcceptHeaderLocaleResolver {
8. **private** Locale myLocal;
9. **public** Locale resolveLocale(HttpServletRequest request) {
10. **return** myLocal;
11. }
12. **public** **void** setLocale(HttpServletRequest request, HttpServletResponse response, Locale locale) {
13. myLocal = locale;
14. }
16. }

然后在action-servlet.xml里的设置为

Xml代码  [收藏代码](javascript:void())

1. **<bean** id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource"**>**
2. **<property** name="basename" value="message" **/>**
3. **</bean>**
4. **<bean** id="myViewController" class="org.springframework.web.servlet.mvc.UrlFilenameViewController" **/>**
5. **<bean** id="filenameController" class="org.springframework.web.servlet.mvc.UrlFilenameViewController" **/>**
6. **<bean** id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"**>**
7. **<property** name="interceptors" ref="localeChangeInterceptor"**/>**
8. **<property** name="mappings"**>**
9. **<value>**
10. chinese.do=filenameController
11. us.do=filenameController
12. **</value>**
13. **</property>**
14. **</bean>**
15. **<bean** id="defaultHandlerMapping" class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" **/>**
16. **<bean** id="localeChangeInterceptor" class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" **/>**
17. **<bean** id="localeResolver" class="org.springframework.web.servlet.i18n.MyAcceptHeaderLocaleResolver"**/>**
18. **<bean** id="viewResolver"
19. class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
20. **<property** name="viewClass"
21. value="org.springframework.web.servlet.view.JstlView" **/>**
22. **<property** name="prefix" value="/WEB-INF/jsp/" **/>**
23. **<property** name="suffix" value=".jsp" **/>**
24. **</bean>**

## 基于SpringMVC国际化资源配置方式。

1.首先需要在spring-mvc-servlet.xml 中配置拦截器；

|  |  |
| --- | --- |
| 1 | <bean id="localeChangeInterceptor" class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" /> |

2.如果要浏览器自适应语言方式（cookie方式）：还需要在spring-mvn-servlet.xml 中配置：

2.1

[?](http://my.oschina.net/Rayn/blog/467291)

|  |  |
| --- | --- |
| 1 | <bean id="cookieLocaleResolver" class="org.springframework.web.servlet.i18n.CookieLocaleResolver"/> |

2.2.如果是需要在界面上进行切换（session方式）；则需要配置：

[?](http://my.oschina.net/Rayn/blog/467291)

|  |  |
| --- | --- |
| 1  2  3 | <bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver">          <property name="defaultLocale" value="en\_US"/>      </bean> |

3.最后在applicationContext.xml 中配置关于资源文件

[?](http://my.oschina.net/Rayn/blog/467291)

|  |  |
| --- | --- |
| 1  2  3  4 | <bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">          <property name="basename" value="message"/>          <property name="useCodeAsDefaultMessage" value="true"/>      </bean> |

4.在拦截器中配置local ，代码如下：

[?](http://my.oschina.net/Rayn/blog/467291)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <mvc:interceptors>            <!--国际化资源切换（根据请求参数中的locale参数自动切换）-->          <mvc:interceptor>              <mvc:mapping path="/\*\*"/>              <ref bean="localeChangeInterceptor"/>          </mvc:interceptor>      </mvc:interceptors> |

5.具体的message\_zh\_CN.properties, message\_en\_US.properties ,等文件，放置于classpath 下。