08-SpringMVC基于注解使用：国际化

1、通过浏览器语言设置国际化化

在日常工作中，如果你的网站需要给不同语言地区的人进行查看，此时就需要使用国际化的基本操作，springmvc的国际化操作比较容易。

login.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html>
<head>
    <title>Title</title>
</head>
<body>
<h1><fmt:message key="welcomeinfo"/></h1>
<form action="login" method="post" >
    <fmt:message key="username"/>: <input type="text" name="username"/><br><br>
    <fmt:message key="password"/>: <input type="password" name="password"/><br><br>
    <input type="submit" value="<fmt:message key="loginBtn"/>"/>
</form>
</body>
</html>
```

I18nController.java

```
package cn.tulingxueyuan.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
```

```java
public class I18nController {

    @RequestMapping("i18n")
    public String i18n(){
        return "login";
    }
}
```

login_en_US.properties

```
welcomeinfo=welcome to tulingxueyuan.cn
username=USERNAME
password=PASSWORD
loginBtn=LOGIN
```

login_zh_CN.properties

```
welcomeinfo=欢迎进入图灵教育
username=用户名
password=密码
loginBtn=登录
```

springmvc.xml

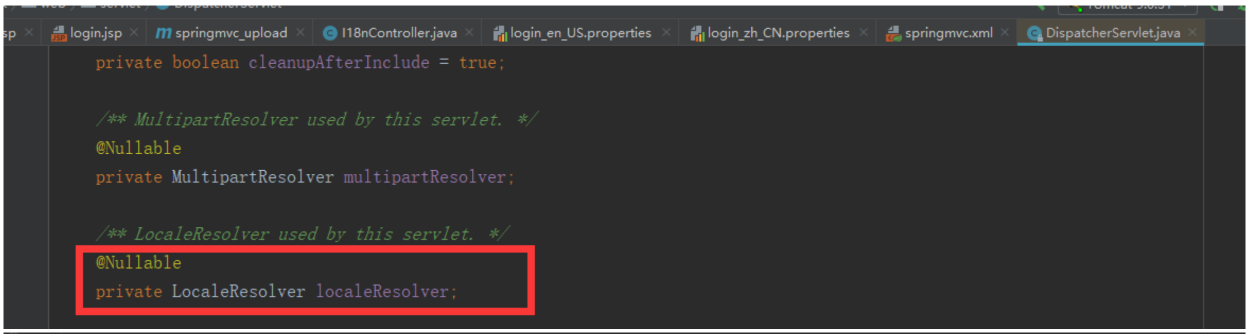
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:mvc="http://www.springframework.org/schema/mvc"

      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springfr
   <mvc:default-servlet-handler></mvc:default-servlet-handler>
   <mvc:annotation-driven></mvc:annotation-driven>
   <context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>

   <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
       <property name="prefix" value="/WEB-INF/page/"></property>
       <property name="suffix" value=".jsp"></property>
   </bean>
   <bean id="messageSource" class="org.springframework.context.support.ResourceBundleMess
       <property name="basename" value="login"></property>
```

```
    </bean>
</beans>
```

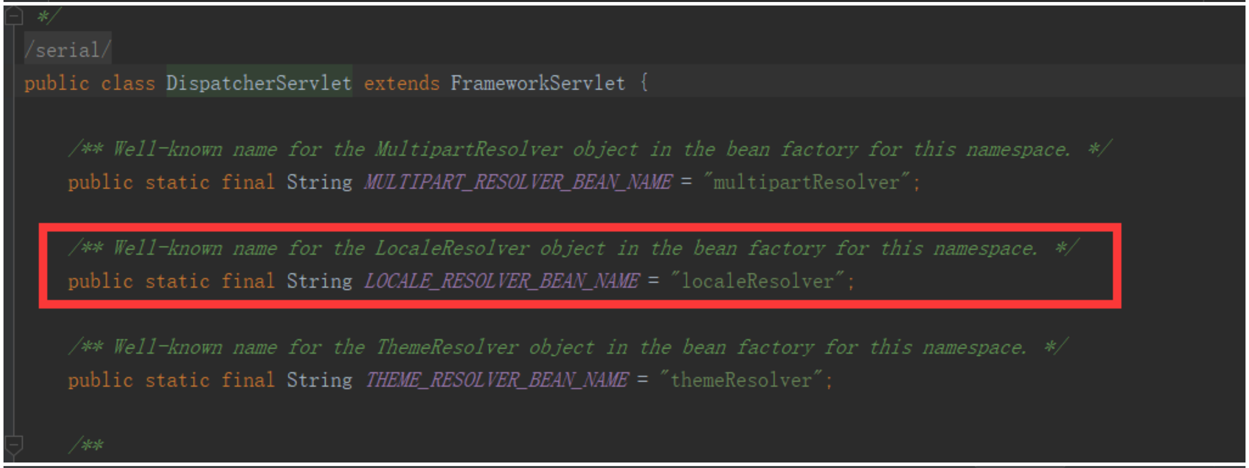
其实SpringMVC中国际化的处理非常简单，就是按照浏览器所带来的语言信息决定的。

- 默认情况下，SpringMVC 根据Accept-Language参数判断客户端的本地化类型。
- 当接受到请求时，SpringMVC 会在上下文中查找一个本地化解析器(LocalResolver)， 找到后使用它获取请求所对应的本地化类型信息：`Locale locale = request.getLocale();//获取浏览器的区域信息`

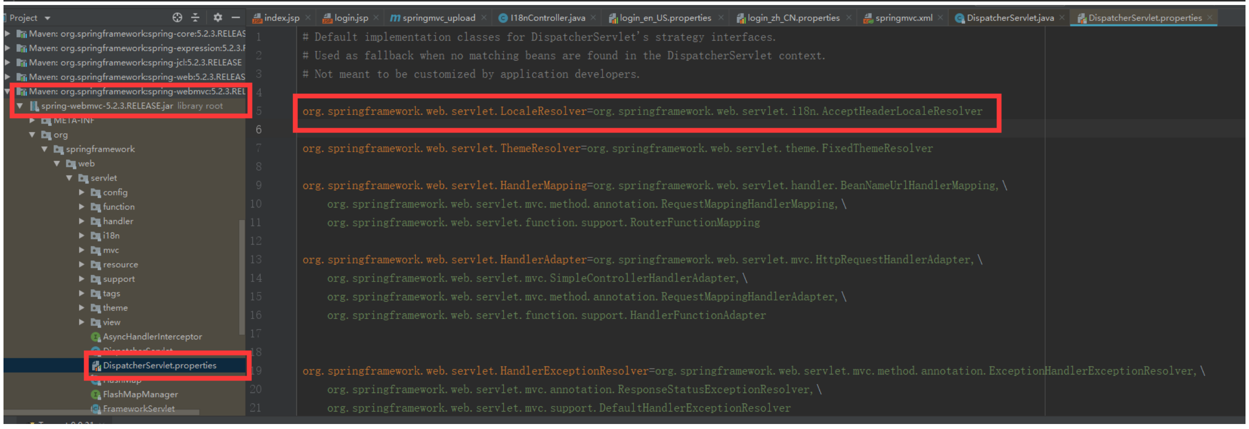
源码：



```java
private boolean cleanupAfterInclude = true;

/** MultipartResolver used by this servlet. */
@Nullable
private MultipartResolver multipartResolver;

/** LocaleResolver used by this servlet. */
@Nullable
private LocaleResolver localeResolver;
```

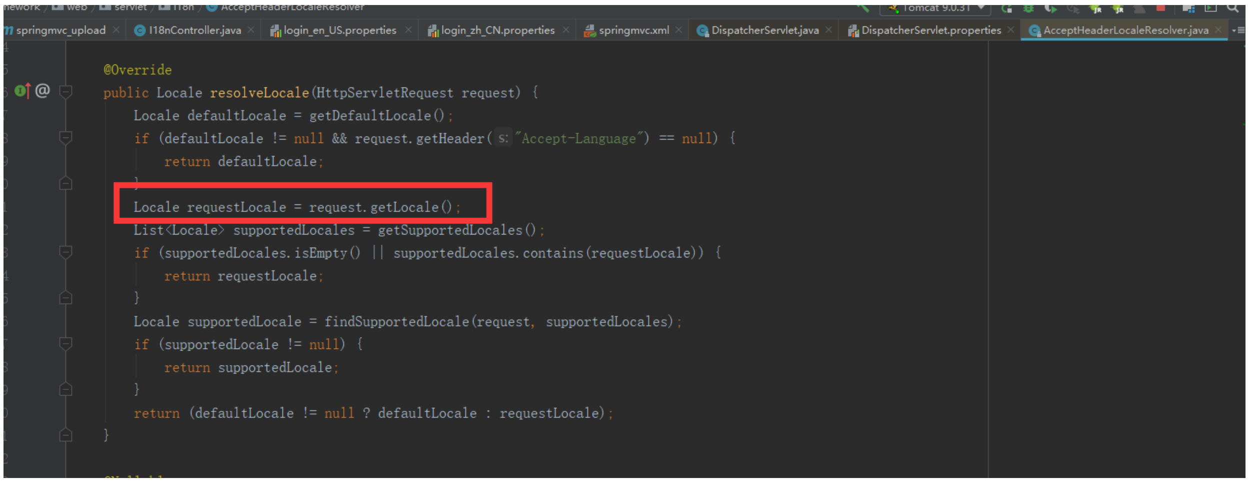


```java
 */
/serial/
public class DispatcherServlet extends FrameworkServlet {

    /** Well-known name for the MultipartResolver object in the bean factory for this namespace. */
    public static final String MULTIPART_RESOLVER_BEAN_NAME = "multipartResolver";

    /** Well-known name for the LocaleResolver object in the bean factory for this namespace. */
    public static final String LOCALE_RESOLVER_BEAN_NAME = "localeResolver";

    /** Well-known name for the ThemeResolver object in the bean factory for this namespace. */
    public static final String THEME_RESOLVER_BEAN_NAME = "themeResolver";

    /**
```



```properties
# Default implementation classes for DispatcherServlet's strategy interfaces.
# Used as fallback when no matching beans are found in the DispatcherServlet context.
# Not meant to be customized by application developers.

org.springframework.web.servlet.LocaleResolver=org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver

org.springframework.web.servlet.ThemeResolver=org.springframework.web.servlet.theme.FixedThemeResolver

org.springframework.web.servlet.HandlerMapping=org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping,\
    org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping,\
    org.springframework.web.servlet.function.support.RouterFunctionMapping

org.springframework.web.servlet.HandlerAdapter=org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter,\
    org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter,\
    org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter,\
    org.springframework.web.servlet.function.support.HandlerFunctionAdapter


org.springframework.web.servlet.HandlerExceptionResolver=org.springframework.web.servlet.mvc.method.annotation.ExceptionHandlerExceptionResolver,\
    org.springframework.web.servlet.mvc.annotation.ResponseStatusExceptionResolver,\
    org.springframework.web.servlet.mvc.support.DefaultHandlerExceptionResolver
```

通过图片能够发现，默认调用的是org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver类



```java
@Override
public Locale resolveLocale(HttpServletRequest request) {
    Locale defaultLocale = getDefaultLocale();
    if (defaultLocale != null && request.getHeader( s: "Accept-Language") == null) {
        return defaultLocale;
    }
    Locale requestLocale = request.getLocale();
    List<Locale> supportedLocales = getSupportedLocales();
    if (supportedLocales.isEmpty() || supportedLocales.contains(requestLocale)) {
        return requestLocale;
    }
    Locale supportedLocale = findSupportedLocale(request, supportedLocales);
    if (supportedLocale != null) {
        return supportedLocale;
    }
    return (defaultLocale != null ? defaultLocale : requestLocale);
}
```

2、通过超链接来切换国际化

- SpringMVC还允许装配--个动态更改本地化类型的拦截器，这样通过指定一个请求参数就可以控制单个请求的本地化类型。

login.jsp

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<html>
<head>
   <title>Title</title>
</head>
<body>
<h1><fmt:message key="welcomeinfo"/></h1>
<form action="login" method="post" >
   <fmt:message key="username"/>: <input type="text" name="username"/><br><br>
   <fmt:message key="password"/>: <input type="password" name="password"/><br><br>
   <input type="submit" value="<fmt:message key="loginBtn"/>"/>
   <a href="i18n?locale=zh_CN">中文</a><a href="i18n?locale=en_US">English</a>
</form>
</body>
</html>
```

MyLocaleResolver.java

```java
package cn.tulingxueyuan;

import org.springframework.web.servlet.LocaleResolver;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.Locale;
```

```java

public class MyLocaleResolver implements LocaleResolver {
    /**
      * 解析并返回locale
      * @param request
      * @return
      */
    @Override
    public Locale resolveLocale(HttpServletRequest request) {
        Locale locale = null;
        String localeStr = request.getParameter("locale");
        if(localeStr!=null && ! "".equals(localeStr)){
            locale = new Locale(localeStr.split("_")[0],localeStr.split("_")[1]);
        }else{
            locale = request.getLocale();
        }
        return locale;
    }

    /**
      * 不支持设置locale的信息
      * @param request
      * @param response
      * @param locale
      */
    @Override
    public void setLocale(HttpServletRequest request, HttpServletResponse response, Local
        throw new UnsupportedOperationException(
                "Cannot change HTTP accept header - use a different locale resolution stra
    }
}
```

springmvc.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"

```

```xml
      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springfr
<mvc:default-servlet-handler></mvc:default-servlet-handler>
<mvc:annotation-driven></mvc:annotation-driven>
<context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <property name="prefix" value="/WEB-INF/page/"></property>
   <property name="suffix" value=".jsp"></property>
</bean>
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessage
   <property name="basename" value="login"></property>
</bean>
   <!--配置区域信息解析器-->
   <bean id="localeResolver" class="cn.tulingxueyuan.MyLocaleResolver"></bean>
</beans>
```

除了可以自定义区域信息解析器之外，我们还可以使用SpringMVC中自带的SessionLocaleResolver:
I18nController.java

```java
package cn.tulingxueyuan.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

import javax.servlet.http.HttpSession;
import java.util.Locale;

@Controller
public class I18nController {

   @Autowired
   private MessageSource messageSource;

   @RequestMapping("i18n")
   public String i18n(@RequestParam(value = "locale",defaultValue = "zh_CN") String loca

       Locale l = null;
       if(localeStr!=null && ! "".equals(localeStr)){
           l = new Locale(localeStr.split("_")[0],localeStr.split("_")[1]);
      }else{
           l = locale;
      }
       session.setAttribute(SessionLocaleResolver.class.getName() + ".LOCALE",l);
       return "login";
  }
}
```

springmvc.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:mvc="http://www.springframework.org/schema/mvc"

      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springf
<mvc:default-servlet-handler></mvc:default-servlet-handler>
<mvc:annotation-driven></mvc:annotation-driven>
<context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <property name="prefix" value="/WEB-INF/page/"></property>
   <property name="suffix" value=".jsp"></property>
</bean>
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessag
   <property name="basename" value="login"></property>
</bean>
   <!--配置区域信息解析器-->
<!--    <bean id="localeResolver" class="cn.tulingxueyuan.MyLocaleResolver"></bean>-->
   <bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleRes
</beans>
```

使用LocaleChangeInterceptor来实现国际化：

springmvc.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"

       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springfr
<mvc:default-servlet-handler></mvc:default-servlet-handler>
<mvc:annotation-driven></mvc:annotation-driven>
<context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <property name="prefix" value="/WEB-INF/page/"></property>
   <property name="suffix" value=".jsp"></property>
</bean>

<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessage
   <property name="basename" value="login"></property>
</bean>
<!--配置区域信息解析器-->
<!--   <bean id="localeResolver" class="cn.tulingxueyuan.MyLocaleResolver"></bean>-->
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolv
   <mvc:interceptors>
       <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"></bean
   </mvc:interceptors>
</beans>
```

I18nController.java

```java
package cn.tulingxueyuan.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

import javax.servlet.http.HttpSession;
import java.util.Locale;
```

```java

@Controller
public class I18nController {

    @Autowired
    private MessageSource messageSource;

    @RequestMapping("i18n")
    public String i18n(@RequestParam(value = "locale",defaultValue = "zh_CN") String loca

//        Locale l = null;
//        if(localeStr!=null && ! "".equals(localeStr)){
//            l = new Locale(localeStr.split("_")[0],localeStr.split("_")[1]);
//        }else{
//            l = locale;
//        }
//        session.setAttribute(SessionLocaleResolver.class.getName() + ".LOCALE",l);
        return "login";
    }
}
```

3、国际化类型转换和验证失败的信息

- **required**：必要的参数不存在。如 @RequestParam(“param1”)

标注了一个入参，但是该参数不存在

- **typeMismatch**：在数据绑定时，发生数据类型不匹配的问，如：

```
typeMismatch.user.birthday= Date format is erro
```

- **methodInvocation**：Spring MVC 在调用处理方法时发生了错误
- **验证失败**：如果是JRS303验证的信息： key的前缀为：注解+对象.属性 。如：

```
NotEmpty.user.username=Username must input!
Length.user.username=Username length must 3-12!
Past.user.birthday=birthday must before today!
```

4、国际化代码中的内容

1.第一种方法 在Handler方法参数中加入Locale参数，注入ResourceBundleMessageSource 对象

```
messageSource.getMessage(code, args,locale);
```

2.第二种方法 或者使用定义工具类：

```
package com.ns.utils;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.support.ResourceBundleMessageSource;

 /**
  * 国际化帮助类
  * @author Administrator
  *
  */
public class I18nMessageUtil {

   private static ResourceBundleMessageSource messageSource;

   private static HttpServletRequest request;

   /**
    * 获取国际化资源属性
    * @param code
    * @param args
    * @return
    */
   public static String getMessage(String code,String... args){
      return messageSource.getMessage(code, args,request.getLocale());
   }
   @Autowired
   public void setMessageSource(ResourceBundleMessageSource messageSource) {
      this.messageSource = messageSource;
   }
   @Autowired
   public void setRequest(HttpServletRequest request) {
        this.request = request;
    }

}

```