11、国际化

笔记本： spring
创建时间： 2022/4/4 17:24
作者： 雷丰阳

1、写好国际化资源文件
2、让Spring的ResourceBundleMessageSource管理国际化资源文件

```
<bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">
        <property name="basename" value="loginpage/login"></property>
    </bean>
```

3、直接去页面取值
4、现象：是按照浏览器带来语言信息决定；
Locale locale = request.getLocale();//获取到浏览器的区域信息

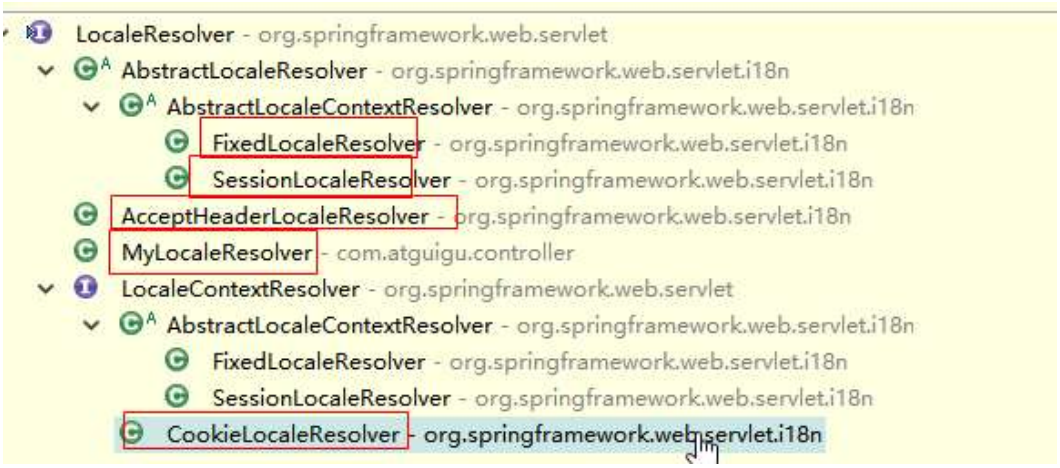
SpringMVC中区域信息是由区域信息解析器得到的；

```
private LocaleResolver localeResolver;
```

默认会用一个`AcceptHeaderLocaleResolver`
所有用到区域信息的地方，都是用到`AcceptHeaderLocaleResolver`获取的

```
Locale locale = this.localeResolver.resolveLocale(request);
```

```
@Override
    public Locale resolveLocale(HttpServletRequest request) {
        return request.getLocale();
    }
```



点击链接切换国际化；（国际化信息是要能改变的）；
AcceptHeaderLocaleResolver：使用请求头的区域信息

```
@Override
```

```
    public Locale resolveLocale(HttpServletRequest request) {
        return request.getLocale();
    }

    @Override
    public void setLocale(HttpServletRequest request, HttpServletResponse
response, Locale locale) {
        throw new UnsupportedOperationException(
                "Cannot change HTTP accept header - use a different locale
resolution strategy");
    }
```

FixedLocaleResolver：使用系统默认的区域信息

```
@Override
    public Locale resolveLocale(HttpServletRequest request) {
        Locale locale = getDefaultLocale();
        if (locale == null) {
            locale = Locale.getDefault();
        }
        return locale;
    }

    @Override
    public LocaleContext resolveLocaleContext(HttpServletRequest request) {
        return new TimeZoneAwareLocaleContext() {
            @Override
            public Locale getLocale() {
                return getDefaultLocale();
            }
            @Override
            public TimeZone getTimeZone() {
                return getDefaultTimeZone();
            }
        };
    }

    @Override
    public void setLocaleContext(HttpServletRequest request,
HttpServletResponse response, LocaleContext localeContext) {
        throw new UnsupportedOperationException("Cannot change fixed locale
- use a different locale resolution strategy");
    }
```

SessionLocaleResolver：区域信息是从session中获取。

可以根据请求参数创建一个locale对象，把他放在session中；

```
@Override
    public Locale resolveLocale(HttpServletRequest request) {
        Locale locale = (Locale) WebUtils.getSessionAttribute(request,
LOCALE_SESSION_ATTRIBUTE_NAME);
        if (locale == null) {
            locale = determineDefaultLocale(request);
        }
        return locale;
    }
```

CookieLocaleResolver

```
@Override
    public Locale resolveLocale(HttpServletRequest request) {
```

```
        parseLocaleCookieIfNecessary(request);
        return (Locale) request.getAttribute(LOCALE_REQUEST_ATTRIBUTE_NAME);
    }
```
