09 Spring IoC容器ApplicationContext如何实现国际化

更新时间：2020-08-10 14:43:24

> 人生太短，要干的事太多，我要争分夺秒。——爱迪生

背景

国际化是指让产品或是程序在无需做出改变的情况下，就能够适应不同语言和地区的需要， internationalization 单词太长，又被简称为 i18n（取头取尾中间有 18 个字母）。

我们知道 ApplicationContext 实现了 MessageSource 接口来实现国家化 i18n 的，我们来看看吧！

ApplicationContext 是如何实现国际化的？

本章节建议先展示国际化的效果（通过国际化可以实现的功能），再通过配置代码去实现。

在 Java 中实现国际化主要是借助一个工具类 ResourceBundle ，核心的思想就是 ，对不同的语言环境提供一个不同的资源文件 。Spring 提供了 ResourceBundleMessageSource 来封装 ResourceBundle 来完成国际化。

实例

能动手就别仅仅动眼，尝试来做一个示例：

1.在 Resources 目录下建立 locale 目录（名字任意取），并创建 messages 开头的不同 locale 的属性文件。

2.配置 ResourceBundleMessageSource 的 bean 使之生效。

可以通过 XML 或者 annotation（注解） 来实现。

XML 方式：

```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:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-
    <!--
    <bean id="bean1" class="com.davidwang456.test.Employee">
      <property name="name" value="Rakesh" />
      <property name="age" value="20" />
    </bean>
    -->
    <bean id="messageSource"
        class="org.springframework.context.support.ResourceBundleMessageSource">
        <property name="basename">
            <value>locale\messages</value>
        </property>
    </bean>
</beans>
```

其中 basename 是指 locale 目录下 messages 开头的文件。

注解方式：

```java
package com.davidwang456.test;
import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ResourceBundleMessageSource;
@Configuration
public class MessageSourceConfig {
    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
        messageSource.setBasenames("locale/messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }
}
```


测试程序

```java
public static void main(String[] args) {
    MessageSource messageSource=new ClassPathXmlApplicationContext("beans.xml");
    String username_us=messageSource.getMessage("userName",null,Locale.US);
    String username_chinese=messageSource.getMessage("userName",null,Locale.CHINESE);
    System.out.println("chinese:"+username_chinese);
    System.out.println("english:"+username_us);
}
```

QA1：

可能会有人说，上面的示例都是单个 properties 文件，如果一个 locale 下有多个属性文件，改如何做呢？

其实很简单，我们来看示例：

```
<beans>
    <bean id="messageSource"
            class="org.springframework.context.support.ResourceBundleMessageSource">
        <property name="basenames">
            <list>
                <value>format</value>
                <value>exceptions</value>
                <value>windows</value>
            </list>
        </property>
    </bean>
</beans>
```

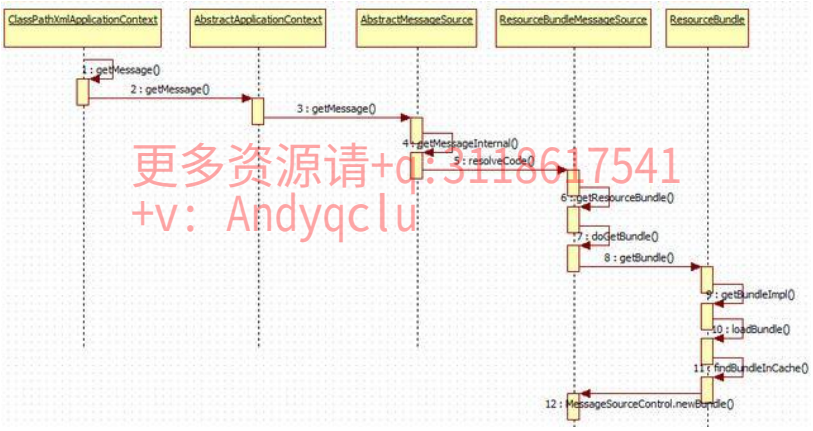
QA2：

还有人会问，你上面的获取指定 locale 的，每次都要带上 locale，支持不同 locale 看着很费劲，其实还可以使用全局变量，除了全家变量外还可以通过设置 jvm 参数，设置 locale，如下示例：

```
set JAVA_ARGS=-Duser.language=fr -Duser.region=FR %JAVA_ARGS%
```

工作原理

通过上面的程序一步步调试，可以看到：



我们来看看 Properties 文件要如何加载的？ResourceBundleMessageSource 的内部类 MessageSourceControl#newBundle() 方法：

```java
private class MessageSourceControl extends ResourceBundle.Control {

    @Override
    @Nullable
    public ResourceBundle newBundle(String baseName, Locale locale, String format, ClassLoader loader, boolean reload)
            throws IllegalAccessException, InstantiationException, IOException {

        // Special handling of default encoding
        if (format.equals("java.properties")) {
            String bundleName = toBundleName(baseName, locale);
            final String resourceName = toResourceName(bundleName, "properties");
            final ClassLoader classLoader = loader;
            final boolean reloadFlag = reload;
            InputStream inputStream;
            try {
                inputStream = AccessController.doPrivileged((PrivilegedExceptionAction<InputStream>) () -> {
                    InputStream is = null;
                    if (reloadFlag) {
                        URL url = classLoader.getResource(resourceName);
                        if (url != null) {
                            URLConnection connection = url.openConnection();
                            if (connection != null) {
                                connection.setUseCaches(false);
                                is = connection.getInputStream();
                            }
                        }
                    }
                    else {
                        is = classLoader.getResourceAsStream(resourceName);
                    }
                    return is;
                });
            }
            catch (PrivilegedActionException ex) {
                throw (IOException) ex.getException();
            }
            if (inputStream != null) {
                String encoding = getDefaultEncoding();
                if (encoding != null) {
                    try (InputStreamReader bundleReader = new InputStreamReader(inputStream, encoding)) {
                        return loadBundle(bundleReader);
                    }
                }
                else {
                    try (InputStream bundleStream = inputStream) {
                        return loadBundle(bundleStream);
                    }
                }
            }
            else {
                return null;
            }
        }
        else {
            // Delegate handling of "java.class" format to standard Control
            return super.newBundle(baseName, locale, format, loader, reload);
        }
    }
```


加载时如果可以找到响应的资源则进行加载，否则创建一个默认的 locale。

第一步的加载使用 ResourceBundleMessageSource.java 的 loadBundle() 完成。

```java
/**
 * Load a property-based resource bundle from the given reader.
 * <p>This will be called in case of a {@link #setDefaultEncoding "defaultEncoding"},
 * including {@link ResourceBundleMessageSource}'s default ISO-8859-1 encoding.
 * Note that this method can only be called with a {@code ResourceBundle.Control}:
 * When running on the JDK 9+ module path where such control handles are not
 * supported, any overrides in custom subclasses will effectively get ignored.
 * <p>The default implementation returns a {@link PropertyResourceBundle}.
 * @param reader the reader for the target resource
 * @return the fully loaded bundle
 * @throws IOException in case of I/O failure
 * @since 4.2
 * @see #loadBundle(InputStream)
 * @see PropertyResourceBundle#PropertyResourceBundle(Reader)
 */
protected ResourceBundle loadBundle(Reader reader) throws IOException {
    return new PropertyResourceBundle(reader);
}

/**
 * Load a property-based resource bundle from the given input stream,
 * picking up the default properties encoding on JDK 9+.
 * <p>This will only be called with {@link #setDefaultEncoding "defaultEncoding"}
 * set to {@code null}, explicitly enforcing the platform default encoding
 * (which is UTF-8 with a ISO-8859-1 fallback on JDK 9+ but configurable
 * through the "java.util.PropertyResourceBundle.encoding" system property).
 * Note that this method can only be called with a {@code ResourceBundle.Control}:
 * When running on the JDK 9+ module path where such control handles are not
 * supported, any overrides in custom subclasses will effectively get ignored.
 * <p>The default implementation returns a {@link PropertyResourceBundle}.
 * @param inputStream the input stream for the target resource
 * @return the fully loaded bundle
 * @throws IOException in case of I/O failure
 * @since 5.1
 * @see #loadBundle(Reader)
 * @see PropertyResourceBundle#PropertyResourceBundle(InputStream)
 */
protected ResourceBundle loadBundle(InputStream inputStream) throws IOException {
    return new PropertyResourceBundle(inputStream);
}
```

读取属性 Properties 资源有两种形式，一种使用 Reader，一种使用 InputStream 方式。

总结：

spring applicationContext 国际化有三要素：**ResourceBundle**，**MessageFormat**，**ResourceBundle.Control**。其背后的原理是根据 locale 来加载响应 locale 的属性文件。

根据 locale 加载属性文件遵循 RFC-4647 规则，查找是根据最优匹配原则，先语言后地区。故如果一个语言仅仅在一个地区有使用的话或者多个地区使用的语言习惯一致的话，只要标识语言就可以了；如果一个语言在多个地区使用，且使用的语言习惯有迥异，如汉字中的简体字和繁体字，那么就需要标明语言和地区了。

}

08 青出于蓝而胜于蓝之揭秘 Spring容器ApplicationContext

10 扩展之Spring MVC中如何实现国际化i18n