# 06. 国际化（SEUG）

## 定义Locale

com.bstek.dorado.core.resource.LocaleResolver接口用于告知Dorado系统当前应使用什么地区和语种，Dorado提供的默认配置如下:

```
<bean id="dorado.localeResolver"
class="com.bstek.dorado.view.resource.SpringLocaleResolverAdapter">
 <property name="springLocaleResolver">
  <bean class="org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver" />
 </property>
</bean>
```

该配置通过Spring中的org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver来确定地区和语种。您也可以通过自定义com.bstek.dorado.core.resource.LocaleResolver的实现类来改变原有的逻辑，新的实现类只要通过下面的方式配置到 home:context.xml 既可生效。

```
<bean id="dorado.localeResolver" class="xxx.MyLocaleResolver"/>
```

在自定义的LocaleResolver中实现resolveLocale方法:

```
public Locale resolveLocale() throws Exception {
 //创建并返回符合条件的java.util.Locale
}
```

## 资源文件命名的规则

资源文件的基本命名规则如下:

```
主文件名[.Locale].properties
```

假设当前系统的Locale是zh_CN，那么Dorado会首先查找带有.zh_CN.properties后缀的资源文件，如果系统中并不存在该文件Dorado会进一步查找只带有.properties后缀的资源文件。

## 公有国际化资源

公有国际化资源是指那些整个系统的各个功能模块重用的国际化资源，公有国际化资源与私有国际化资源的区别主要体现在缓存管理。出于节省系统内存的考虑，我们建议您只把那些确实可能被重用的资源项放入公有国际化资源文件。

公有国际化资源文件应被放置在搜索路径（SearchPath）下，系统中可以存在1到多个SearchPath。添加SearchPath的方式如下:

```
<bean parent="dorado.globalResourceSearchPathRegister">
 <property name="searchPath" value="home:resources/" />
</bean>
```

每一组资源文件被称为一个资源束（Bundle），即那些主文件名相同但Locale不同的资源文件。主文件名即被认为是资源束（Bundle）的名称（

BundleName）。例如当我们要使用一个名为Test的Bundle时，Dorado将依次到各SearchPath中根据BundleName（即Test）和通过Locale Resolver确定的Locale寻找匹配的资源文件，并直接使用找到的第一个。

- Default是一种特殊的BundleName，当我们需要访问名为Default的Bundle中资源项时可以不指定BundleName。
- 资源文件并非只能放在SearchPath的根路径中，您也可以把它们放到子目录中。例如当你把一个Test.zh_CN.properties放置在SearchPath对应目录的名为core的子目录中时，它的BundleName就应该是core.Test。

每个Bundle中可以包含很多个资源项，一个资源项通常就是一段文本。另外，我们也可以在这段文本中植入一些参数，例如：

```
newMessageNotify=您收到了%d条新的消息!
```

关于此处参数的具体用法请参阅：http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html中的 String.format() 方法。

# 私有国际化资源

为了让国际化的使用更加方便，Dorado允许下面的几种对象拥有自己的私有国际化资源：

- Model配置文件 － 与Model配置文件存放在同一路径，且主文件名相同的资源文件被Dorado视为Model配置文件的私有国际化资源。
- View配置文件 － 与View配置文件存放在同一路径，且主文件名相同的资源文件被Dorado视为View配置文件的私有国际化资源。
- JavaBean － 与Java文件存放在同一路径，且主文件名相同的资源文件被Dorado视为该JavaBean的私有国际化资源。

> 在Dorado推荐的开发方式中我们常常把一个View相关的拦截器方法、监听方法都定义在一个与View配置文件同名同位置的JavaBean中。在这种情况下，该View配置文件和JavaVean可以共享同一份私有国际化资源。这是完全符合通常的理解习惯的，同时也恰恰体现了这种开发方式的优势。

对于Model配置文件和View配置文件的私有国际化资源，Dorado还为他们提供一种非常重要的功能——资源的自动注入，关于此功能介绍见本文后面的“国际化资源的自动注入”一节。

# 使用国际化资源

使用国际化资源主要有以下两种方式——EL表达式、ResourceManager

## EL表达式

EL表达式主要用于在Model或View的配置文件中引用一段国际化资源。以这一段EL表达式为例：

```
${res.PageTitle}
```

假设我们是在View配置文件中定义了上面的这样一段EL表达式。Dorado在处理时会首先到该View配置文件的私有国际化资源中查找名为“PageTitle”的资源项，如果找到则直接使用它的值。如果没找到，Dorado将继续到名为“Default”的国际化资源束查找名为“PageTitle”的资源项（我们在前面提到过Default是一种默认的BundleName）。

下面的两种写法与上面的写法完全的等价。

```
${res["PageTitle"]}
${res.get("PageTitle")}
```

下面是一个稍微复杂了一点的例子：

```
${res["Test/PageTitle"]}
```

对于这段表达式，Dorado在处理时会首先到该View配置文件的私有国际化资源中查找名为“Test/PageTitle”的资源项，如果找到则直接使用它的值。如果没找到，Dorado将继续到名为“Test”的国际化资源束查找名为“PageTitle”的资源项。

> 从这里我们可以知道，Dorado利用“/”来分隔bundleName和key的。这也提醒了我们，不要在定义资源项的key时使用“/”字符，否则非常容易一起混淆。

如果我们要使用的资源项是带有参数的，可以按照下面的方法来定义EL表达式：

```
${res.get("NewMessageNotify", 5)} // 传入一个参数
${res.get("Test/ResourceWith3Args", "arg1", 17, "arg3")} // 传入3个参数
```

如果资源文件是放在SearchPath的子目录中，例如当你把一个Test.zh_CN.properties放置在SearchPath对应目录的名为core的子目录中时，它的BundleName就应该是core.Test。可以按照下面的方法来定义EL表达式：

```
${res["core.Test/PageTitle"]}
```

对于这段表达式，Dorado在处理时会到SearchPath的core子目录中名为“Test”的国际化资源束查找名为“PageTitle”的资源项。

### ResourceManager

com.bstek.dorado.core.resource.ResourceManager通常用于访问JavaBean的私有国际化资源，下面是一段简单的例子：

```
@Component
public class MyBean {
 @DataProvider
 public Collection<Employee> findEmployeesByNamePattern(String pattern) throws Exception {
  if (StringUtils.isEmpty(pattern)) {
   // 获得当前Class的私有国际化资源束
   ResourceManager resourceManager = ResourceManagerUtils.get(getClass());
   throw new IllegalArgumentException(resourceManager.getString("PatternUndefined"));
  }
  ... ...
 }
}
```

ResourceManager中处理path的过程与EL表达式中的过程一致，这里不做累述。更的具体用法请参考Dorado7的JavaDoc。

> 需要特别指出的是，EL表达式和ResourceManager并不只是简单的访问私有国际化资源，通过他们都可以访问公有国际化资源。只是在查找顺序方面，私有国际化资源的优先级高于公有国际化资源。

## 国际化资源的自动注入

想象一下，如果我们的某个View中有一个包含了50个PropertyDef的DataType，而这50个PropertyDef的caption和tip属性都需要进行国际化。那么我们可能需要在这个View中填写100此EL表达式，这么做无疑是令人崩溃的一件事！正是考虑到了这样的使用场景，Dorrado为国际化资源提供了主动注入的功能。

自动注入功能目前适用于Model和View这两种文件，基本的做法是只要按照特定的规则在私有国际化资源文件中定义资源项，这些资源字符串就会在运行时自动的被注入到相应的属性中，不需要额外的定义EL表达式去引用。

所以支持自动注入的资源项都必须以“#”作为其键值的开头。以View配置文件为例，假设我们在其私有资源文件定义了如下的资源项：

```
\#buttonSave=保存
```

由于#在Java的.properties文件中表示注释，所以我们需要在第一个“#”字符之前增加“\”。Dorado在遇到这样的资源项时会把“#”后面的内容认作View中某控件的id，并且自动将“保存”设置到该控件（实质为Button）的caption属性中。

看到这里您可能会产生一个疑问，为什么“保存”被注入到了caption属性中而不是其他的属性？原因是在进行自动注入时，如果没有显示的指定的属性名，Dorado默认会按照下面的规则来确定属性，首先查找控件是否存在caption属性，如何存在则直接设置caption属性，如果没有则进一步查找label属性，进而查找title属性。此查找规则可以被定义在具体Class上的Annotation改变，定义这种Annotation的方法此处不表。

- 在注入的过程中如果Dorado发现对象的目前属性的内容不是空，那么Dorado会跳过该资源的注入。也就是说直接定义在View配置文件的内容的优先级是高于资源自动注入的。
- 另外还有一个需要提及的规则是，#后面的内容其实并不仅仅表示控件的id，也可以是DataType的名称。具体代表控件还是DataType，将有Dorado在运行时自动判断。
- 如果您需要为View配置文件中的View对象本身注入国际化资源，那么可以通过“#view”完成，因为View对象不允许我们为其定义id属性。

下面的更多的资源注入的实例：

```
\#buttonSave.tip=保存当前记录  <-- 注入到id为buttonSave的按钮的tip属性
\#view=代码维护     <-- 注入到View的title属性
\#Employee.#address=地址  <--
查找名为Employee的DataType，并注入到其中address属性描述的caption属性中
\#Employee.#address.tip=请输入您的地址 <-- 注入到上面属性描述的tip属性中
\#gridEmployee.#comment=备注  <--
查找id为gridEmployee的DataGrid，并注入到其中comment列的caption属性中
```

# 引入外部资源

在某些系统中，用户可能不希望以.properties文件的形式来管理国际化资源。例如将所有的国际化资源保存是数据库中就是一种比较常见的场景。对于这种场景，我们可以对Dorado的公有国际化资源机制进行一些扩展来引入这些数据库中的资源项。

要引入外部的资源，首先您需要定义一个com.bstek.dorado.core.resource.ResourceBundle的实现类，代码大致如下：

```
public class DBResourceBundle implements ResourceBundle {
 public String getString(String key, Object... args) throws Exception {
  // 这里您要做的就是读取自己的数据库，并返回与key匹配的资源项的内容。
  // 强烈建议您在此处尽心一些缓存方面的处理，而不是每次调用该方法都读取一次数据库。
  // 否则这里很可能会成为整个系统的性能瓶颈。
 }
}
```

然后您需要扩展GlobalResourceBundleManager，代码大致如下：

```
public class MyGlobalResourceBundleManager extends DefaultGlobalResourceBundleManager {
 @Override
 protected ResourceBundle doGetBundle(String bundleName, Locale locale)
   throws Exception {
  if ("DB".equals(bundleName)) {
   //
如果您在此处拦截了Default，那么今后访问来自DBResourceBundle中的资源时，您就可以不必指定bundleName
。
   // 如果您在此处拦截了所有的bundleName，那么系统原有的公有国际化资源就相当于被完全屏蔽了，
   // 那样的话建议您直接继承GlobalResourceBundleManagerSupport类
   return new DBResourceBundle();
  } else {
   return super.doGetBundle(bundleName, locale);
  }
 }
}
```

最后将您自己定义的GlobalResourceBundleManager配置到Dorado中即告完成，例如在 home:context.xml 中添加如下配置：

```
<bean id="dorado.globalResourceBundleManager" class="xxx.MyGlobalResourceBundleManager">
 <property name="cache" ref="dorado.globalResourceCache" />
</bean>
```

如果按照本例设定bundleName为"DB"，则EL表达式使用时的参考范例：

```
${res["DB/PageTitle"]}
```

ResourceManager的用法：

```
@Component
public class MyBean {
 @DataProvider
 public Collection<Employee> findEmployeesByNamePattern(String pattern) throws Exception {
  if (StringUtils.isEmpty(pattern)) {
   // 获得DBResourceBundle国际化资源束
   ResourceManager resourceManager = ResourceManagerUtils.get("DB");
   throw new IllegalArgumentException(resourceManager.getString("PatternUndefined"));
  }
  ... ...
 }
}
```

# 定制Dorado客户端的国际化

此部分的内容，事实上与上面的内容基本完全无关。Dorado的客户端所使用的国际化与上面的方式几乎是两个完全不同的体系。这里我们只简单的介绍一下如何为Dorado的客户端定义新的国际化资源或者覆盖其中的一部分国际化资源。

Dorado客户端的国际化资源文件默认全部存放在classpath:dorado/resources/i18n这个资源路径下，系统默认值提供了两组默认的资源，即英文资源和zh_CN资源。如果要为Dorado提供一组新的资源文件，例如tw_CN。最简单的方法就是直接在resources/i18n这个资源路径下添加一组tw_CN.properties文件。注意这里指的并不是把这些资源文件重新压缩进dorado-core-7.xxxx.jar中。

某些时候，我们要做可能仅仅是替换系统默认提供的资源中某几个资源项。

例如在resources/i18n/core.zh_CN.properties中存在一个名为UnknownEvent的资源项，您可能不习惯我们官方提供的翻译，而想把它改成另外一中说法。此时您可以自己新建一个同名文件，放在任意的位置。例如放在 home:resources/client/
中。例如您自己定义的core.zh_CN.properties的内容如下：

home:resources/client/core.zh_CN.properties

```
dorado.core.UnknownEvent=未声明的事件"{0}"。
```

然后，将这个文件配置到 home:context.xml 中：

```
<bean parent="dorado.clientI18NFileRegister">
 <property name="packageName" value="core" />
 <property name="path" value="home:resources/client/core" />
</bean>
```