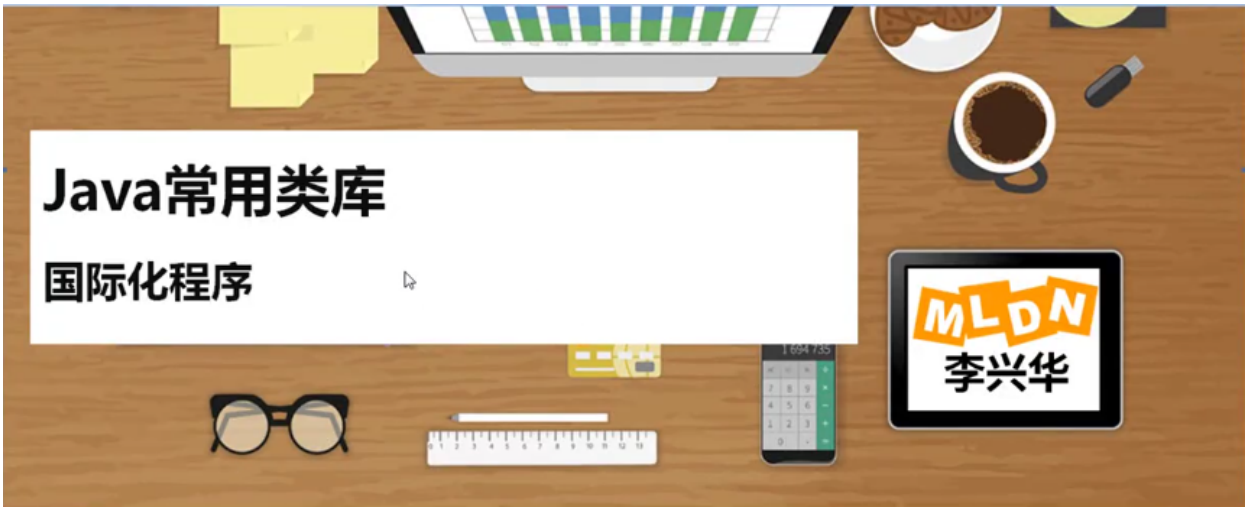


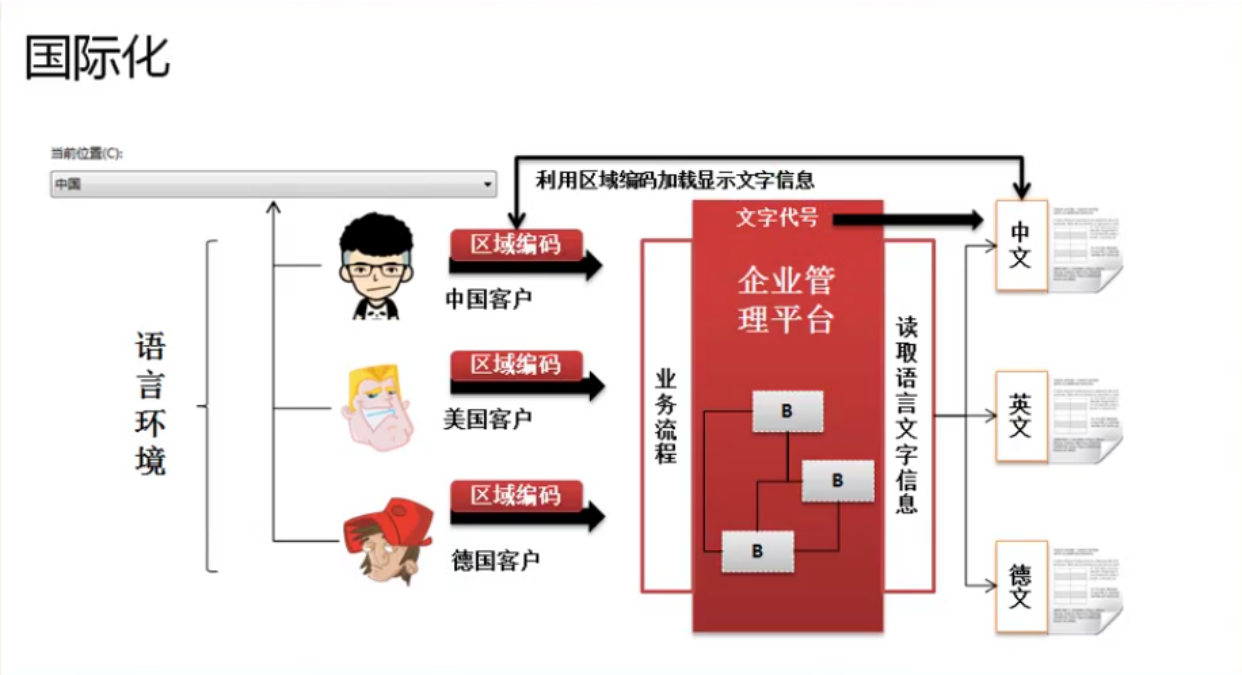
博客： https://www.cnblogs.com/HOsystem/p/14116443.html

2、具体内容

所谓的国际化的程序指的是同一个程序代码可以根据不同的国家实现不同的语言描述，但是程序处理的核心业务是相同的。

■国际化问题简介

现在假设有一款世界都认可的企业管理平台，那么这个企业的老板决定将这个产品推广到全世界各个上市的公司，于是这些公司可以来自于：中国、美国、德国，那么在这样的情况下，首先要考虑的问题是什么呢？



通过分析之后可以发现，如果要想实现国际化的程序开发，那么要解决的问题就在于以下两点：

·如何可以定义保存文字的文件信息；

·如何可以根据不同的区域语言的编码读取指定的资源信息；

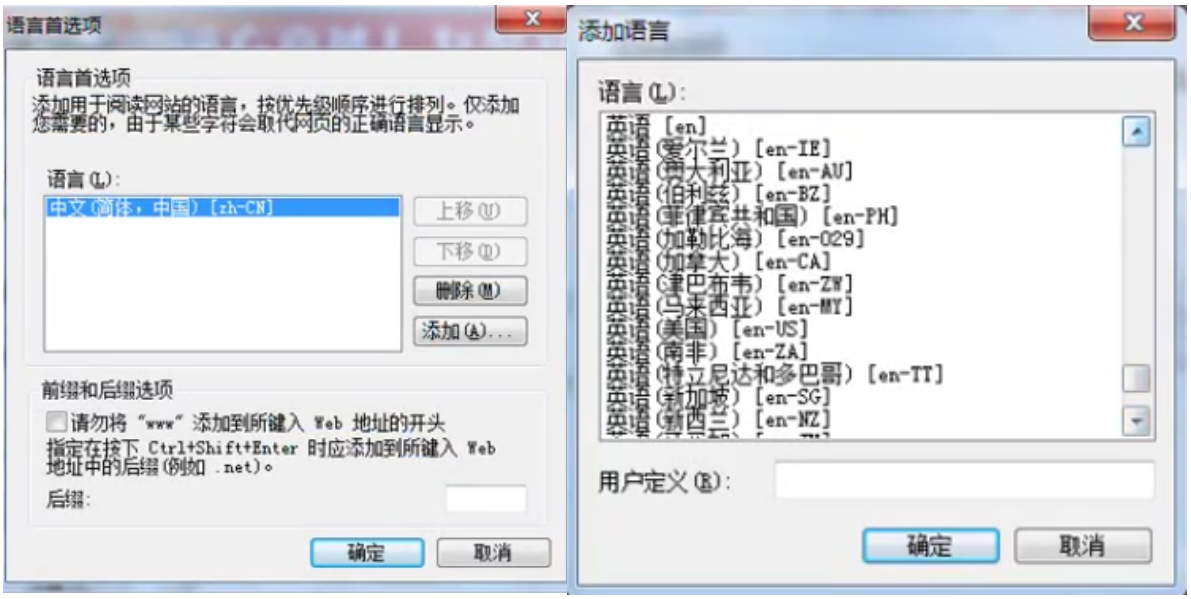
■Locale类

通过分析可以发现，如果要想实现国际化，那么首先需要解决的就是不同国家用户的区域和语言编码问题，而在java.util包里面提供有一个专门描述区域和语言编码的类：Locale，而后主要可以使用Locale类中的两个构造方法进行实例化：

·构造方法：public Locale(String language)；

·构造方法：public Locale(String language,String country)；

此时需要的是国家和语言的代码，而中文的代码：zh_CN、美国英语的代码：en_US，对于这些区域和语言的编码最简单的获得方式就是通过IE浏览器。



范例：实例化Locale类对象

```
import java.util.Locale;

public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Locale loc = new Locale("zh", "CN");	// 中文环境
		System.out.println(loc);
	}
}
```

如果说现在要想自动获得当前的运行环境，那么现在就可以利用Locale类本身默认的方式进行实例化：

·读取本地默认环境：public static Locale getDefault()；

```
import java.util.Locale;

public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Locale loc = Locale.getDefault() ;  // 获取默认环境
		System.out.println(loc);
```

```
    }
}
```

在实际的开发过程之中，很多人可能并不关心国家和语言的编码，所以为了简化开发，Locale也将世界上一些比较著名的国家的编码设置为了常量。

```
import java.util.Locale;

public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
        Locale loc = Locale.CHINA ;
        System.out.println(loc);
    }
}
```

使用常量的优势在于可以避免一些区域编码信息的繁琐。

■读取资源文件：ResourceBundle

现在已经准备好了资源文件，那么随后就需要进行资源文件的读取操作了，而读取资源文件主要依靠的是java.util.ResourceBundle类完成，此类定义如下：

```
public abstract class ResourceBundle extends Object
```

ResourceBundle是一个抽象类，如果说现在要想进行此类对象的实例化可以直接利用该类中提供的一个static方法完成：

·获取ResourceBundle对象：public static final ResourceBundle getBundle (String baseName);

|- baseName：描述的是资源文件的名称，但是没有后缀（**cn.mldn.message.Messages**）

·根据key读取资源内容：public final String getString(String key);

范例：使用ResourceBundle类读取内容

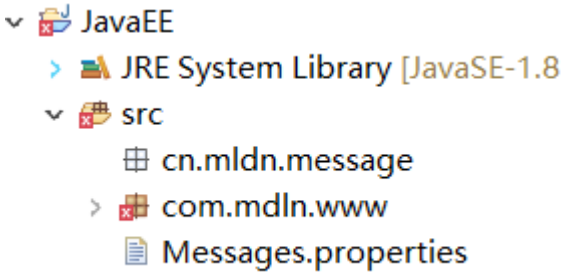
```
import java.util.ResourceBundle;

public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
        ResourceBundle resource = ResourceBundle.getBundle("cn.mldn.message.Messages") ;
        String val = resource.getString("info") ;
        System.out.println(val);
    }

}
```

```
//cn.mldn.message.Messages.properties
//要创建packagecn.mldn.message然后创建文件Messages.properties
info=欢迎小强同学光临指教！！！
```

如果资源没有放在包里面，则直接编写资源名称即可。




```
ResourceBundle resource =
ResourceBundle.getBundle("Messages") ;
String val = resource.getString("info") ;
```

在进行资源读取的时候数据的key一定要存在，如果不存在出现如下异常信息：“Exception in thread "main" java.util.**MissingResourceException**: Can't find resource for bundle java.util.PropertyResourceBundle, key infos”；

■实现国际化程序

现在国际化程序的实现前期准备已经全部完成了，也就是说依靠资源文件、Locale、ResourceBundle类就可以实现国际化的处理操作，那么下面来进行国际化的程序实现（核心关键：读取资源信息）。

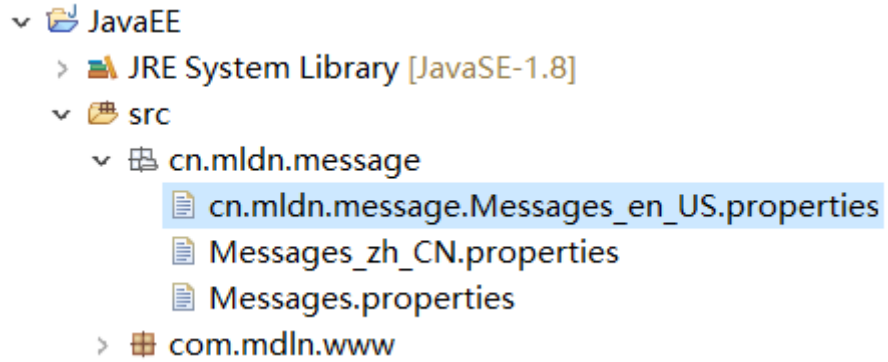
1、在CLASSPATH下建立：**cn.mldn.message.Messages_*zh_CN*.**properties（中文资源）；

```
info=\u6B22\u8FCE\u4F60\u7684\u8BBF\u95EE\uFF01//info=欢迎你的访问!
```

2、在CLASSPATH下建立：**cn.mldn.message.Messages_*en_US*.**properties（英文资源）；

```
info=welcome!
```

现在加上没有默认的区域的资源文件，一共定义了三个资源：



3、通过程序进行指定区域的资源信息加载

```
import java.util.ResourceBundle;
public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
```

```
        ResourceBundle resource =
ResourceBundle.getBundle("cn.mldn.message.Messages") ;
        String val = resource.getString("info") ;
        System.out.println(val);
    }

}
```

此时在利用ResourceBundle类读取资源的时候并没有设置一个明确的Locale对象，但是发现“Message_zh_CN”文件起作用了，因为这个方法里面默认加载的就是当前本地的Locale的资源：

```
public static final ResourceBundle getBundle(String baseName)
  {
        Class<?> caller = Reflection.getCallerClass();
    return getBundleImpl(baseName, Locale.getDefault(),caller,
                                    getDefaultControl(caller,baseName));
  }
```

4、如果现在有需要也可以修改当前的Locale环境，则可以使用ResourceBundle类中的如下方法：

·获取ResourceBundle：public static final ResourceBundle getBundle(String baseName,**Locale locale**)；

```
import java.util.Locale;
import java.util.ResourceBundle;
public class JavaAPIDemo {
    public static void main(String[] args) throws Exception {
        Locale loc = new Locale("en", "US");
        ResourceBundle resource =
ResourceBundle.getBundle("cn.mldn.message.Messages", loc);
        String val = resource.getString("info");
        System.out.println(val);
    }

}
```

如果现在有指定区域的资源文件存在的时候，那么没有设置区域的资源文件的信息将不会被读取。

读取顺序：读取指定区域的资源文件 > 默认的本地资源 > 公共的资源*（没有区域设置）

■消息格式化

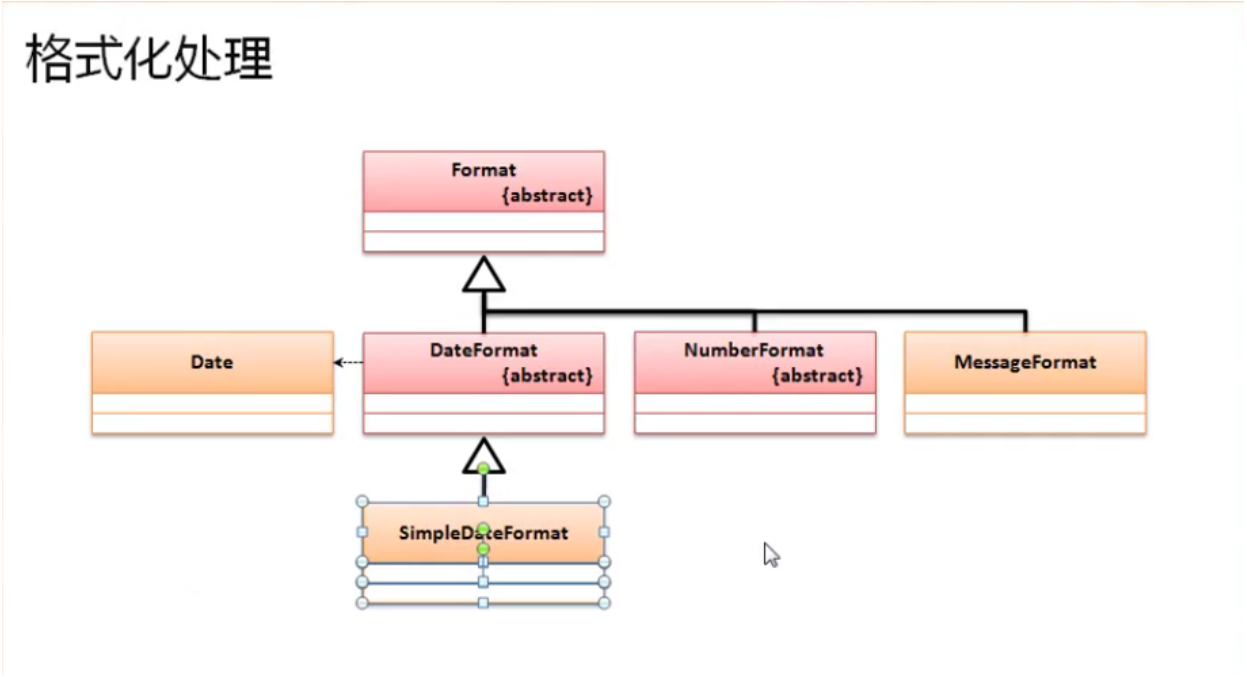
如果说现在某一位用户登录成功了，那么一般都会显示这样的信息“XXX，欢迎你的光临~”，也就是说这个时候会显示用户名，那么此时如果这些内容保存在了资源文件里面，则就需要通过占位符来进行描述，同时对于读取出来的数据也需要进行消息格式化的处理。

范例：修改资源文件

【中文资源文件】 cn.mldn.message.Messages_zh_CN.properties	info=欢迎{0}的访问,当前日期是{1}!
【英文资源文件】 cn.mldn.message.Messages_en_US.properties	info=welcome {0},date:{1}!

如果有需要则可以继续添加“{1}”、“{2}”之类的内容；

此时如果要进行资源读取则会将占位符的信息一起读取出来，所以此时就需要利用MessageFormat类进行格式化处理。



在MessageFormat类中提供有一个格式化文本的方法：public static String format(String pattern, Object...arguments)；

范例：格式化文本实现国际化

```
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.ResourceBundle;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		Locale loc = new Locale("en", "US");
		ResourceBundle resource =
ResourceBundle.getBundle("cn.mldn.message.Messages", loc);
		String val = resource.getString("info");
		System.out.println(MessageFormat.format(val, "mldn", new
SimpleDateFormat("yyyy-MM-dd").format(new Date())));
	}

}
```

如果在日后开发的过程之中见到资源文件里面出现有“{0}”、“{1}”的结构表示的都是占位符，该信息一定都要进行格式化。