# 第 15 章　国际本地化与中文乱码问题

随着国际化的深入，软件产品也应该国际化和本地化，即适合不同国际和地区用户的使用。本章将重点介绍国际化、本地化的概念和实现过程。另外还将介绍在 JSP 开发过程总经常困惑开发者的中文乱码问题。

**本章要点包括以下内容：**

- JSP 程序的国际化以及本地化。
- Web 应用开发过程中的中文乱码问题。

## 15.1　Web应用程序的国际化与本地化

当开发的软件产品需要国际化，在全球范围内进行使用，这时就需要考虑到此软件产品是否能够适应全球不同国家和地区用户。最为简单和直观的要求是语言问题，即所开发的软件产品上显示的文本信息是否能够在本地语言环境下正确的显示。

当然除了语言问题，一个优秀的国际化软件产品还需要能够支持不同格式的日期、时间、货币等其他值的显示，而不需要在本地对软件进行大量的修改，甚至一点都不需要修改。其中在 JDK 中已经向开发者提供了有关国际化和本地化开发的类，开发人员只要灵活利用这些类并能开放出具有国际本地化的程序。在这一点上，JSP 就要比 ASP 与 PHP 动态网页技术更加的优越。

在开始讲解这一小节内容之前，首先了解以下两个常用的术语：

- i18n：它表示的是 internationalization（国际化）单词，由于 i 和 n 单词之间间隔 18 个字符，所以就简称为 i18n。“国际化”是为了使开发的应用程序能够适应不同的语言环境，而不需要做任何的程序修改所采取的设计措施。
- l10n：它表示的是 localization（本地化）单词。由于 l 和 n 字母之间间隔 10 个字符，所以简称为 l10n。“本地化”是为了应用程序能够在一个特定的语言环境下进行使用，从而加入本地特殊化格式以及调用本地语言文本的一个过程，本地化是一个过程。

经过以上介绍之后，读者应该认识到要想开发出一个能够适应全球不同国家和地区用户的软件产品，就需要经过以上介绍的国际化和本地化两个阶段。下面主要介绍日期和多国语言的两个国际化和本地化实例，通过该实例加深读者对这两个概念的了解。当然，有关国际化和本地化的内容还很多，但是方法和思想都一样，读者可以根据这两点触类旁通。

### 15.1.1　日期的国际化

Java 已经提供了对软件国际化和本地化的支持。在 JDK 包含了一些实现国际化和本地化的类，开发者可以直接使用这些类快速、方便地开发过出国际化和本地化的软件产品。其中用于本地化的基类为 Local 类，它的一个实例化对象代表一个特定地理区域。而其他类则使用这个类的实例对象来格式化日期、数字、本地化字符串和应用程序中的其他对象。

本小节介绍日期的国际化问题，不同国家或者地区，它们的日期显示方式是不一样的。为了解决这

个问题，Java 提供了一个 java.text.DateFormat 类来实现日期的国际化。具体看下面的一个实例代码，该实例使用 DateFormat 类来格式化不同地区的日期显示方式：

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.text.DateFormat,java.util.Locale,java.util.Date" %>
<%
    Date date = new Date();
    DateFormat localeDate = DateFormat.getDateInstance(DateFormat.LONG);
    DateFormat usDate = DateFormat.getDateInstance(DateFormat.LONG,Locale.US);
    DateFormat germanDate = DateFormat.getDateInstance(DateFormat.LONG,Locale.GERMAN);
    DateFormat italyDate = DateFormat.getDateInstance(DateFormat.LONG,Locale.ITALY);
    DateFormat frenchDate = DateFormat.getDateInstance(DateFormat.LONG,Locale.FRENCH);
%>
<html>
<head><title>日期的国际化</title></head>
<body>
本地日期：<%=localeDate.format(date)%><br>
美国日期：<%=usDate.format(date)%><br>
德国日期：<%=germanDate.format(date)%><br>
意大利日期：<%=italyDate.format(date)%><br>
法国日期：<%=frenchDate.format(date)%><br>
</body>
</html>
```

程序说明：

（1）该页面代码第一行设置 page 指令中的 contentType 属性，其中指定文档输出格式为 text/html，使用的字符编码为 gb2312。第二行通过 import 属性引入该页面将使用到的各类文件。引入方式可以通过一个 import 指定一个类文件，也可以通过一个 import 属性来指定多个类文件，这些类文件之间使用逗号隔开。

（2）Locale 类：所有提供本地化支持的 Java 类都使用一个叫做 java.util.Locale 的类。该类的一个实例表示一个特定的地理、政治或文化上的区域。该区域是由一个国家代码和一个语言代码的组合指定的。根据用户的语言和当地风俗的不同由 Locale 实例来决定执行不同任务（通过指定参数值）。构造 Locale 实例的构造函数一一国际代码和语言代码作为参数，HTTP 也使用这两个标准来定义语言和国家代码。例如，java.util.Locale useLocale = new Locale(“en”,”US”)，就是一个本地化了的 Locale 实例，它表示的是英语语言以及美国地区的语言环境。

（3）从代码中可以看出使用 java.text.DateFormat 类国际化日期是相当简单的，首先调用 DateFormat 类中的 getDateInstance()方法，并且向方法中传递日期显示格式以及地区参数，从而获得一个 DateFormat 实例，例如：

```
DateFormat usDate = DateFormat.getDateInstance(DateFormat.LONG,Locale.US);
```

然后再使用 usDate 实例来显示当前的日期：

```
美国日期：<%=usDate.format(date)%><br>
```

其中 Locale 类中已经定义很多静态常量，例如 US 就代表“en_US”、CHINA 就表示“zh_CN”。

（4）DateFormat.getDateInstance()方法的第二参数就是使用 Locale 类中的地区常量来实现日期显示格式的本地化。当第二个参数不设置时，程序将使用浏览器默认的本地化设置。

打开浏览器，运行以上编写的 JSP 程序，运行效果如图 15.1 所示。


图 15.1　日期的国际化

## 15.1.2　实现多国语言版本的JSP实例

要使一个软件产品具有多国语言版本（即支持多国语言），Java 提供了一个资源类 java.util.ResourceBundle 来实现。其实 ResourceBundle 只是一个抽象类，它有两个子类：ListResourceBundle 和 PropertyResourceBundel。前一个子类是需要将编写的不同国家语言信息放置在对象类中，下面将实例演示；后一个子类是需要编写不同国家语言的文件（后缀为.properties）来存放语言信息，这部分内容读者可以在讲解国际化标签以及 Struts 时了解到。例如，存放中文的 Res_zh_CN.properties 和存放英文的 Res_en_US.properties 资源文件。

接下来将使用 ResourceBundle 抽象类的一个 ListResourceBundle 之类来实现一个多国语言版本的 JSP 应用程序。该实例需要创建的文件如下：

- selectLanguage.jsp：选择所要显示的语言。
- multiLanguage.jsp：显示不同语言信息的主页面。
- Res_zh_CN.java：中文资源类，包名为 cn.com.resource。
- Res_en_US.java：英文资源类，包名为 cn.com.resource。

当需要支持更多国家语言时，可以创建更多相应的资源类。

### 15.1.2.1　中文资源类 Res_zh_CN.java

该创建的资源类必须要继承 ListResourceBundle 类，其中详细代码如下：

```
package cn.com.resource;
import java.util.ListResourceBundle;
public class Res_zh_CN extends ListResourceBundle{
    //创建对应的中文文本信息
    static final Object[][] contents = new String[][]{
        {"version","这是中文版"},{"title","中文版本"},{"index","首页"},{"news","新闻"},
        {"life","生活"},{"sports","体育"},{"entertainment","娱乐"}
    };
    protected Object[][] getContents() {
        // TODO Auto-generated method stub
        return contents;
    }
}
```

代码说明：该代码首先必须要继承 ListResourceBundle 之类，并定义 contents 数组变量来存放某一语言的所有信息（由一个关键字来标识）。另外，还需要定义一个 getContents()方法来返回 contents 变量值。

#### 15.1.2.2．英文资源类 Res_en_US.java

和上一个类文件一样，区别只是该对象类存放英文信息，详细代码如下：

```
package cn.com.resource;
import java.util.ListResourceBundle;
public class Res_en_US extends ListResourceBundle{
    //创建对应的英文文本信息
    static final Object[][] contents = new String[][]{
        {"version","This is Engilish Version"},{"title","English Version"},{"index","index"},{"news","news"},
        {"life","life"},{"sports","sports"},{"entertainment","entertainment"}
    };
    protected Object[][] getContents() {
        // TODO Auto-generated method stub
        return null;
    }
}
```

**注意：这里的创建的资源类，它的命名必须由“基本类名_语言代码_地区代码组成。只有这样的命名方式，才能根据“基本类名”查找到相应的资源类。基本类名可以任意取名，语言代码是指定的代码（例如，cn 代码中文，en 代表英文），地区代码也是确定的一组代码（例如 CN 表示中国大陆，US 表示美国地区），具体看** multiLanguage.jsp **页面程序。**

#### 15.1.2.3．语言选择页面 selectLanguage.jsp

在该页面的下拉框中选择你所希望使用的语言版本，详细代码如下：

```
<%@ page contentType="text/html;charset=GBK"%>
<html>
<head><title>选择语言</title></head>
<body>
<form action="multiLanguage.jsp" method="get">
    <select name="language">
        <option value="default">默认</option>
        <option value="chinese">中文</option>
        <option value="english">English</option>
  </select>
  <input type="submit" value="提交">
</form>
</body>
</html>
```

代码说明：选择好语言之后，单击“提交”按钮，由 multiLanuguage.jsp 页面来显示不同版本语言。如果用户选择是“默认”选项，下面的 multiLanguage.jsp 页面将根据用户当前浏览器有关语言设置来显示不同的语言。该页面显示效果如图 15.2 所示。


图 15.2　选择语言

#### 15.1.2.4．显示多国语言的页面 multiLanguage.jsp

该页面显示的是一个导航栏信息，可以根据用户在 selectLanguage.jsp 页面中选择的不同语言来显示不同语言版本信息。详细代码如下：

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.ResourceBundle,java.util.Locale"%>
<%
    String language = request.getParameter("language");
    //默认的语言，浏览器会根据请求中所包含的 Accept language 来辨别默认的为何种语言
    ResourceBundle res = ResourceBundle.getBundle("cn.com.zzb.eshopping.resource.Res");
    //中文版本的显示
    if(language.equals("chinese")){
        res = ResourceBundle.getBundle("cn.com.zzb.eshopping.resource.Res",new Locale("zh","CN"));
    }
    //英文版本的显示
    if(language.equals("english")){
        res = ResourceBundle.getBundle("cn.com.zzb.eshopping.resource.Res",new Locale("en","US"));
    }
%>
<html>
<head><title><%=res.getString("title")%></title></head>
<body>
<%=res.getString("version")%>
<hr>
<%=res.getString("index")%> | <%=res.getString("news")%> | <%=res.getString("life")%> |
<%=res.getString("sports")%> | <%=res.getString("entertainment")%>
</body>
</html>
```

（1）浏览器的请求中包括一个 Accept-Language 首部，该首部包含了一个用户或几个用户想要使用的语言。选择语言的方式由浏览器的配置决定。如果你指定了多种语言，可以以逗号分隔的列表加入首部，每种语言要么仅用语言代码表示，要么用短划线(-)连接语言代码和国家代码。

例：Accept-Language:en-US,en,sv

该例表示第一选择是美国英语，然后是任何类型的英语，最后是瑞典语。由于本实例浏览器默认的是中文显示，即首部设置为 Accept-Language:zh_CN，根据这个设置 ResourceBundle 会查找到 Res_zh_CN.java 资源类，效果如图 15.3 所示。

（2）ResourceBundle 类使用 getBundle 方法，并通过资源类的基本类名以及 Locale 类给定的语言和地区类型，从而获取到相应的资源类。

当用户没有使用 Locale 指定语言和地区类型时，系统会根据浏览器发送请求中所包含的 Accept-Language 来确定使用哪个资源类。

当选择英文时，显示的效果如图 15.4 所示。

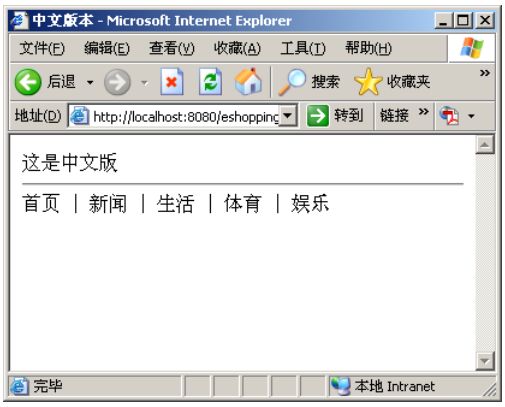


图 15.3　默认语言页面

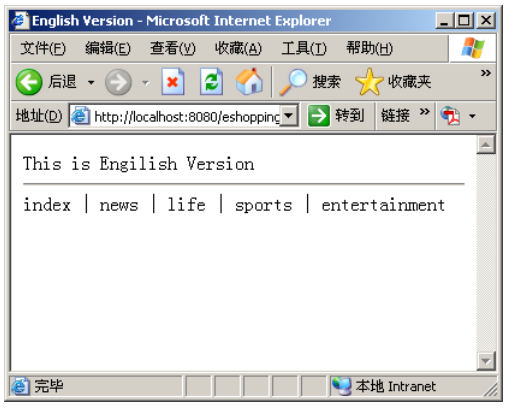


图 15.4　英文页面

至此，本 JSP 实例程序就实现了国际化和本地化的功能，读者一定会发现学习这部分知识并不困难，但是它很重要，特别是在这国际化程度很高的时代，而开发的软件产品又想走向国际市场。

在后面讲解国际化标签时，这部分知识还会提及，那时候读者将会了解到后面一种实现国际化和本地化的方法，即使用资源文件。

# 15.2　中文乱码问题

中文乱码处理在应用系统开发过程中，一直是一个让开发者非常头疼的问题，特别是开发数据库系统。一般开发者（特别是初学者）都要为这类乱码问题花费大量时间，这样问题的根源是在于使用字符集标准不统一。各国都设立了自己的字符集标准，例如，一个英文字符是由一个字节来表示的，而中文是由两个字节表示，那么一个字节表示字符的字符集标准就不能正确显示中文字符（出现乱码）。另外，很多软件对国际化的支持标准和实现方法也各不相同，因此中文乱码问题已经存在很久，但是至今仍然困惑着很多开发人员。

在介绍如何解决中文乱码问题之前，首先认识一下各种常用的字符集。

## 15.2.1　字符集介绍

### 15.2.1.1　ASCII 码

ASCII 码是最早的字符集，它的全称为：American Standard Code for Information Interchange（美国信息交换标准码）。它使用一个字节（共八位）中的 7 位来表示一个字符（a、b 或者 c 等），二进制的一位是有两种可能的（0 或者 1），所以经过组合总共可以表示 128 个字符（2 的 7 次方）。后来，IBM 充分使用一个字节有八位的空间，把 ASCII 改为由八位表示一个字符，这样可以表示 258 个字符（2 的 8 次方）。

ASCII 码是最早的字符集，而其他所有字符集都是在 ASCII 码的基础上提出来的，例如，gb2312

和 GBK。

#### 15.2.1.2　LATIN1

LATIN1 是英语国家和西欧各国的编码标准，它小而精干。

#### 15.2.1.3　gb2312

gb2312 编码是我国针对汉字最早提出的一套编码标准，它已经得到了国际标准组织及工业标准的广泛认可。这种标准是采用双字节来表示一个中文字符，这样总共可以表示 65536 个字符，但是一级汉字只使用了 3755 个字符，二级汉字使用了 3008 多个，总共最多能表示 6763 个字符，所以还剩下很多空间没有使用。gb2312 主要收录简化汉字及符号、字母、日文假名等共 7445 个图形字符，其中汉字占 6763 个。

#### 15.2.1.4　GBK

随着计算机的广泛使用，有时就需要使用到一些非常偏僻的人名、地名以及古籍字，由于 gb2312 只能表示 6763 个字符，所以 gb2312 这时就显得有点无能为力了。正是由于这些问题的出现，出现了 GBK 编码标准，它是对 gb2312 的扩充，是可以向上兼容的（即使用 gb2312 编码的字符可以在 GBK 编码标准下正确显示）。

GBK 也把常用的繁体字加了进来，所以它使得汉字数扩展到了 20902 个，其编码范围是 0x8140-0xfefe。现在，国际标准组织及工业界也都已经接受了 GBK 标准，用它来替代 gb2312。

最近，我国标准部门又推出 GB18030-2000(GBK2K)编码标准，它是在 GBK 的基础上进一步扩展了汉字，增加了藏、蒙等少数民族的字形。GBK2K 从根本上解决了字位不够，字形不足的问题。

#### 15.2.1.5　UNICODE

UNICODE 字符集编码的诞生就是为了统一全球的字符编码，它是（Universal Multiple-Octet Coded Character Set）通用多八位编码字符集的简称，支持世界上超过 650 种语言的国际字符集。Unicode 允许在同一服务器上混合使用不同语言组的不同语言。它是由一个名为 Unicode 学术学会(Unicode Consortium)的机构制订的字符编码系统，支持现今世界各种不同语言的书面文本的交换、处理及显示。该编码于 1990 年开始研发，1994 年正式公布，最新版本是 2005 年 3 月 31 日的 Unicode4.1.0。Unicode 是一种在计算机上使用的字符编码。它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。

Unicode 标准始终使用十六进制数字，而且在书写时在前面加上前缀“U+”，例如字母“A”的编码为 004116。所以“A”的编码书写为“U+0041”。

Unicode 标准编码的成生就可以避免所要不同字符集的冲突，而导致页面乱码的产生。现在软件开发商通常的做法就是只写一套针对 UNICODE 编码标准的程序，然后把不同语言配上不同的资源文件，使得软件可以国际化，例如 Eclipse 开发工具，汉化的过程就是使用中文资源文件替代原有的文件。

#### 15.2.1.6　UTF-8

UTF-8 是 Unicode 的其中一个使用方式。UTF 是 Unicode Translation Format，即把 Unicode 转做某种格式的意思。UTF-8 便于不同的计算机之间使用网络传输不同语言和编码的文字，使得双字节的 Unicode 能够在现存的处理单字节的系统上正确传输。UTF-8 使用可变长度字节来储存 Unicode 字符，例如 ASCII 字母继续使用 1 字节储存，重音文字、希腊字母或西里尔字母等使用 2 字节来储存，而常用的汉字就要使用 3 字节。辅助平面字符则使用 4 字节。

## 15.2.2　产生乱码的原因

产生中文乱码的原因，最主要就是在开发系统的各个环节或阶段中，使用的字符集不一致或者前后不兼容，从而文本信息在被调用和显示过程中不能正确被下一阶段识别，所以出现了乱码。在开发一个 Web 应用系统时，对中文的正确显示有影响的环节包括：

（1）操作系统环境（Windows、Linux）：Windows XP 已经能够支持 GBK 了。如果使用的是 Linux 或者 UNIX，则需要检查其字符集环境的设置。

（2）数据库环境（MySQL、Oracle）：例如第十章的 10.3 小节，在安装 MySQL 数据库时，会要求设置 MySQL 数据库环境的字符集（在本书的安装中已经设置了 GBK）。除了 MySQL 数据库环境需要设置使用的字符集之后，还需要设置库、表和字段所采用的字符集。但是在 MySQL 中有字符集继承的关系：

- 库的字符集如果没有设置，则会默认使用 MySQL 环境设置的字符集。
- 表的字符集如果没有设置，则会默认使用库设置的字符集。
- 字段的字符集如果没有设置，则会默认使用表设置的字符集。

（3）编程环境（Java、C++）：在 Java 代码中使用 response.setContentType()方法进行设置，例如 response.setContentType("text/html;charset=GBK")。但是 Java 环境已经能够很好的支持中文显示。

（4）中间件环境（Tomcat、Weblogic 和 JBoss）：tomcat 默认编码是 iso-8859-1，可以在 server.xml 进行修改。

（5）JSP 页面编译时：在头部设置 contentType 属性值来设置所采用的字符集。

从而可以看出，开发一个 Web 应用程序，涉及的环节很多，所以特别容易出现乱码等问题。

## 15.2.3　解决方法

在 JSP 环境编码过程中出现的乱码问题主要集中在 tomcat 服务器和 MySQL 数据库中。如果在 Web 应用开发过程中出现乱码，主要解决方法有：

#### 15.2.3.1　在 tomcat 中修改 server.xml

因为 tomcat 默认编码是 iso-8859-1，这里把它改成中文编码。注意黑体字部分。

```
<Connector port="8080" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true" URIEncoding='GBK' />
```

当然还可以使用其他字符集编码，都是一定要记着前后环节使用的字符集要统一或者前后兼容。

#### 15.2.3.2　修改 Java 编程环境的字符集

在 Java 程序中添加如下一段代码来修改 Java 编程环境所采用的字符集：

```
response.setContentType("text/html; charset=GBK");
```

它的作用是让浏览器把 Unicode 字符转换为 GBK 字符。这样页面的内容和浏览器的显示模式都设成了 GBK，就不会乱码了。

#### 15.2.3.3　MySQL 数据库所采用的字符集

MySQL 数据库环境默认字符集是 latin1，但是可以在安装时修改它的默认编码，请查看第二章的 2.3.2 小节关于 MySQL 安装的内容。用代码编写创建数据库表时在后面加 default charset=GBK 指定表的编码，如果使用工具创建数据库或者表，一般会有字符集选择项，设置成和前后环节统一的字符集标准。

#### 15.2.3.4　Jsp 页面编辑时

在 JSP 文件的头部加上<%@ page contentType="text/html;charset=GBK" %>。

**注意：最近一次作者就是因为 charset=GBK 的等号左右多了 2 个空格而出错，主要是因为 c++/java 的编写方式习惯性加了空格，以后大家要注意了。第二，charset 的字母应该全小写，不然会出现乱码。**

#### 15.2.3.5　连接数据库时

```
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/sample_db?user=&password=&useUnicode=true&characterEncoding=GBK");
```

但是这个环节一般是不需要的。

一般情况，开发者不需要对以上环节一一进行设置（因为很多环节已经对中文显示有很好的支持）。首先查看 JSP 页面中是否进行 charset 属性设置，如果还没有解决中文乱码问题，再查看数据表的编码设置是否一致。最后再查看 Tomcat 容器以及其他环节的编码设置。

### 15.2.4　中文乱码的深层解决方法

已知，开发并不需要进行如上的所有设置，才能解决中文乱码问题。现在很多软件产品都能够很好的支持中文显示，在 Web 应用开发过程中，一般做到如下两点就可以避免乱码问题：

（1）在 MySql 数据库安装以及表的创建过程中，注意采用支持中文的字符集（推荐使用 UTF-8，使得国际化成为可能）。

（2）在 JSP 文件的头部添加如下两段代码：

```
<%@ page contentType="text/html;charset=GBK" %>
<META http-equiv=Content-Type content="text/html;charset=GBK">
```

这里采用的字符集为 GBK，当然开发人员也可以选择 gb2312 或者 UTF-8，但是一定要保证前后环节的字符集统一或者前后兼容。

如果这两步设置还不能解决乱码问题，再逐步针对以上介绍的四个步骤进行检查和修改。

最为糟糕的情况，就是进行以上四部修改之后，还不能解决 Tomcat 服务器的中文乱码问题，这时就需要为 Tomcat 服务器添加一个设置字符集的 Filter 类：

```
package filters;                          //所在包，任意取
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.UnavailableException;
public class SetCharacterEncodingFilter implements Filter {
protected String encoding = null;
protected FilterConfig filterConfig = null;
protected boolean ignore = true;
public void destroy() {
    this.encoding = null;
    this.filterConfig = null;
 }
public void doFilter(ServletRequest request, ServletResponse response,FilterChain chain)
                                                    throws IOException, ServletException {
if (ignore || (request.getCharacterEncoding() == null)) {
```

```
String encoding = selectEncoding(request);
    if (encoding != null)
        request.setCharacterEncoding(encoding);
}
chain.doFilter(request, response);
 }
public void init(FilterConfig filterConfig) throws ServletException {
this.filterConfig = filterConfig;
this.encoding = filterConfig.getInitParameter("encoding");
String value = filterConfig.getInitParameter("ignore");
if (value == null)
   this.ignore = true;
else if (value.equalsIgnoreCase("true"))
   this.ignore = true;
   else if (value.equalsIgnoreCase("yes"))
      this.ignore = true;
   else
   this.ignore = false;
   }
protected String selectEncoding(ServletRequest request) {
   return (this.encoding);
   }
}
```

程序说明：该类一定要继承一个 Filter 类，关于过滤器类的概念和创建会在第二十章中重点讲解。然后在 web.xml 文件进行相应的配置：

```
<filter>
<filter-name>Set Character Encoding</filter-name>
<filter-class>filters.SetCharacterEncodingFilter</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>GBK</param-value>
</init-param>
</filter>

<filter-mapping>
<filter-name>Set Character Encoding</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

代码说明：<filter-class>元素项就是设定 SetCharacterEncodingFilter.java 类所存放的具体位置。

## 15.3　本章小结

本章节首先重点介绍了 JSP 程序开发 Web 应用时所遇到的国际化和本地化问题，主要通过两种方法：一种是使用 ListResourceBundle 子类来创建资源类实现国际化；第二种使用 propertyResourceBundle 子类来创建资源配置文件（例如在后面讲解国际化标签和 struts 时，所要讲到的 ResourceMessage.properties 资源配置文件）。除了文本的国际化之外，还包含日期，货币单位等等，但是这些已经得到了 Java 的很多支持，Java 也提供了很多类，开发者只要调用这些就可以实现国际化。接

着介绍了在 Web 开发过程中经常遇到的中文乱码问题，总结了出现乱码问题的原因以及解决方法。读者在实际开发过程中再慢慢加深对以上问题的体会。