# 1、JSP页面乱码

　　这种乱码的原因是没有在页面里指定使用的字符集编码，解决方法：只要在页面开始地方用下面代码指定字符集编码即可。

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>

# 2、数据库乱码

这种乱码会使你插入数据库的中文变成乱码，或者读出显示时也是乱码，解决方法如下：

在数据库连接字符串中加入编码字符集。

String Url="jdbc:mysql://localhost/digitgulf?user=root&password=root&useUnicode=true&characterEncoding=GB2312";

并在页面中使用如下代码：

response.setContentType("text/html;charset=gb2312");

request.setCharacterEncoding("gb2312");

# 3、中文作为参数传递乱码

当我们把一段中文字符作为参数传递给另一页面时，也会出现乱码情况，解决方法如下：

在参数传递时对参数编码，比如

RearshRes.jsp?keywords=" + java.net.URLEncoder.encode(keywords)

然后在接收参数页面使用如下语句接收

　　keywords=new String(request.getParameter("keywords").getBytes("ISO-8859-1"));

以上为现阶段遇到的乱码问题，乱码的核心问题还是字符集编码问题，只要掌握了这一点，一般的乱码问题都可以解决。这是后台的解决方法，不是前台的。

前台代码参数传递：

# 4、创建拦截器

（1）创建一个java类去实现Filter接口。

（2）在doFilter()方法中编写：

request.setCharacterEncoding("gbk");

chain.doFilter(request,response);

（3）修改配置文件web.xml

# 5、Java中对中文字符串的编码和解码

## 1、getBytes(“编码方式”)

如：String str = "中国"

**编码**：byte[] bts = str.getBytes("编码方式");//常用编码方式 gbk、utf-8、gb2312、iso-8859-1等。

**解码**：String b = new String(bts,"解码方式");//解码方式对应常用编码方式。

正常（正常显示的字符串无乱码）解码后的字符串对象可以用任意方式编码。但解码要正常显示，必须用对应的编码方式解码。（对于中文要保证正常显示必须采用中文编码/解码方式）

如 String str = "中国";//这个就是正常显示的字符串

以下任意方式（中文）编码/解码。

String b1 = new String(str.getBytes("gbk"),"gbk");

String b2 = new String(str.getBytes("utf-8"),"utf-8");

String b3 = new String(str.getBytes("gb2312"),"gb2312");

以上三种方式的字符串对象都是采用中文方式编码/解码，所以都会是正常并无乱码。

3、当没采用对应方式解码时（也就是所谓的乱码）怎么转成正常显示而无乱码。

如 String str = "中国";

String b1 =  new String(str.getBytes("gbk"),"gbk");//这里的b1就是采用的gbk的方式解码的。

String b2 = new String(b1.getBytes(),"utf-8");//这里对b1编码后的字节数组重新用utf-8而没有用对应的gbk方式解码，b2将会是乱码的。

以下开始将乱码转成正常显示

String b3 = new String(b2.getBytes("utf-8"),"gbk")//这里编码方式（utf-8）必须用上述b2的解码方式,而解码方式（gbk）必须用b1的解码方式。因为是b2把正常显示的b1解码成乱的，b3不会是乱码的。

在实际应用中出现的乱码，都是这个原因造成的，乱码不是不可逆的，上述步骤就是一个逆转的过程。

在web开发中，tomcat对于传输的字符串都是采用iso-8859-1编码/解码方式。而客户端（浏览器端对于中文都是用gbk或utf-8中文编码/解码方式），所以传到后台都会是乱码的。容器一般都是有处理的，所以中文能正常显示和存储。但有些情况也是会出现乱码的，解决方式如下：

String b = new String(str.getBytes("iso-8859-1"),"客户端的编码/解码方式")//中文解码方式一般用的是utf-8或者gbk。

如：

 String b1 = new String(str.getBytes("iso-8859-1"),"utf-8");

## 2、URLEncoder与URLDecoder

String strTest = "?=abc?中%1&2<3,4>";  
strTest = URLEncoder.encode(strTest, "UTF-8");//编码  
System.out.println(strTest);  
strTest = URLDecoder.decode(strTest,"UTF-8");//解码  
System.out.println(strTest);

结果：

%3F%3Dabc%3F%E4%B8%AD%251%262%3C3%2C4%3E  
?=abc?中%1&2<3,4>

# 5、字符编码：GBK、gb2312与UTF-8的区别

**GB2312编码**大约包含6000多汉字（不包括特殊字符）,编码范围为第一位b0-f7,第二位编码范围为a1-fe(第一位为cf时,第二位为a1-d3),计算一下汉字个数为6762个汉字。当然还有其他的字符。包括控制键和其他字符大约7573个字符编码

**gbk编码**是对gb2312编码的扩充，容纳的汉字更多，但仅仅是扩充，没有质的变化。保留了所有gb2312编码,在此基础上进行编码范围的扩充.容纳(包含特殊字符)共22014个字符编码。

**unicode编码**(也就是UTF编码)：俗称万国码，致力于使用统一的编码准则表达各国的文字。为表达更多的文字，utf-8采用2/3混编的方式。目前容纳的汉字范围小于gbk编码。并且以3字节的方式处理中文，带来了兼容性的问题，原有的gbk,gb2312,gb18030编码文件都不能正常的处理,还有很长的路要走。