# 编码加密

## 绪论

一般而言，网站、页面的编码通常都是UTF-8。如果将url使用iso-8859-1进行编码之后，对于中文是不可逆的。

当使用ISO8859

· String srcChar = "ISO8859-1";

· String dstChar = "UTF-8";

· title = new String(title.getBytes(srcChar),dstChar);

· content = new String(content.getBytes(srcChar),dstChar);

· nickName = new String(nickName.getBytes(srcChar),dstChar);

· linkAddress = new String(linkAddress.getBytes(srcChar),dstChar);

String str = "品牌/系列名称";  
String s = new String(str.getBytes(), "iso-8859-1");  
String de = new String(s.getBytes("iso-8859-1"), "UTF-8");

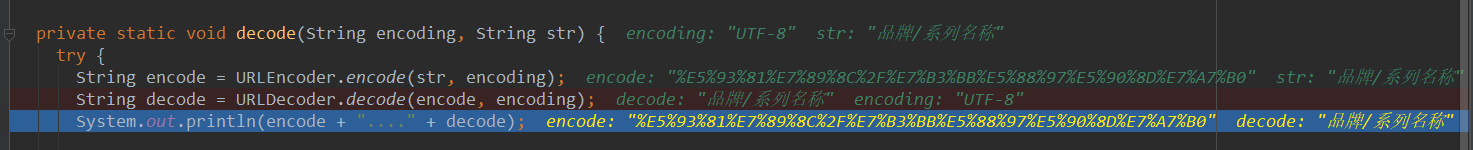
注：UTF-8的流被错误的当成ISO8859-1的流，所以便得到了一个错误的String。才能用这种方式解码，并不是能将ios-8859-1编码格式转化为UTF-8格式。

## 编码格式

### Unicode编码

### UTF-8编码

### URL编码/解码（支持UTF-8和gb2312）



不支持ios8859-1编码

### Base64编码

## Url请求编码

### 设置请求的编码

request.setCharacterEncoding("UTF-8");

## [谈谈对Java中Unicode、编码的理解](http://www.cnblogs.com/newstar/archive/2011/06/13/2079870.html)

     我们经常会遇到编码问题。Java号称国际化的语言，是因为它的class文件采用UTF-8，而JVM运行时使用UTF-16（至于为什么JVM中要采用UTF-16，我没看过 相关的资料，但我猜可能是因为JAVA里面一个字符(char)就是16位的，而UTF-16正是双字节编码），都是unicode的编码。

     unicode 的目标就是能支持世界上所有的字符集，也就是说几乎所有的字符集包含的字符在unicode中都有对应的编码。在unicode中，字符与代码的映射关 系，就是unicode字符集，称为UCS(Unicode Character Set)，每个unicode字符编码称为code point（代码点？）。UTF-8和UTF-16是不同的UCS编码方法，UTF就是UCS Transformation Format。;

     在Java 中，String的getBytes()方法就是对特定的字符串(unicode)按照给定的字符集进行编码（encode），new String()则可以按照某个字符集将字节流转换回unicode（decode）。Java里面的每一个String都是unicode编码。

     再来看页面，如果不做特殊处理，Form的提交就按照页面的ContentType设置中的字符集进行编码转换，发送到后台，后台必须利用req.setCharacterEncoding来指定参数的编码格式(不同的应用服务器应有不同的指定方式)，才能正确解码。

     Java 里面的encode和decode都是相对于unicode而言的，encode的意思是将char[] --> XXX Encoding byte[]，decode就是由XXX Encoding byte[] --> char[]。平常，当我们说“将GBK编码转换为UTF-8编码”的时候，实际的意思就是：GBK Encoding byte[] --> UTF-8 Encoding byte[]，这种转换只有在需要用byte[]传输数据的时候才有意义，否则便是毫无意义的。

      首先要说明的一点是：Java中的String对象就是一个unicode编码的字符串。

      但是，我们通常会听到有人说：“我们需要将String由ISO-8859-1转换为GBK编码”，这又是怎么回事呢？实际上，我们并不是要“将 一个由ISO-8859-1编码的String转换为GBK编码的String”，反复说明的是，JAVA中的String都是unicode编码的，所以不存在“ISO- 8859-1编码的String”或“GBK编码的String”这样的说法。而需要转换的唯一的原因是String进行了错误的编码。我们经常会碰到由ISO-8859- 1转换为诸如GBK/UTF-8等等这样的需求。所谓的转换过程是：String --> byte[] -->String。  
也许 你非常清楚这个过程的代码：new String(text.getBytes("ISO-8859-1"),"GBK")。但是，要真正理解起来并不是那么简单。表面上看似乎很容易理解， 不就是将text String对象按照ISO-8859-1的方式编码为byte[]然后再把它按照GBK的方式转换为String吗？但是这句代码很容易会被误解为： “将text String由ISO-8859-1转换为GBK编码”，这种说法是错误的。难道你见过用这样的代码：new String(text.getBytes("GBK"),"UTF-8")来对String进行编码转换的吗？

      之所以你会经常看到new String(text.getBytes("ISO-8859-1"),"GBK")这句代码，是因为一个GBK的字节流被错误地以ISO-8859- 1的方式转换为String（unicode）了！发生这种情况最普遍的地方是一个GBK编码的网页向后台提交数据的时候，就有可能会看到这句代码的出 现。GBK的流被错误的当成ISO8859-1的流，所以便得到了一个错误的String。由于ISO8859-1是单字节编码，所以每个字节被按照原样 转换为String，也就是说，虽然这是一个错误的转换，但编码没有改变，所以我们仍然有机会把编码转换回来！所以那句经典的new String(text.getBytes("ISO-8859-1"),"GBK")便出现了。

如果系统误以为是其它编码格式，就有可能再也转换不回来了，因为编码转换并不是负负得正那么简单的