前言1：由于计算机只识“数”，因此人们必须告诉计算机哪个数字来代表哪个特定字符，例如65代表字母‘A’，66代表字母‘B’，以此类推

前言2：当时的计算机普遍使用8比特字节作为最小的存储和处理单元，加之当时用到的字符也很少，26个大小写英文字母还有数字再加上其他常用符号，也不到100个，因此使用7个比特位就可以高效的存储和处理ASCII码，剩下最高位1比特被用作一些通讯系统的奇偶校验。

前言3：一个字节有八个位，00000000，每个位有0和1两种情况，所以一个字节所表示信息有2^8次方即256个

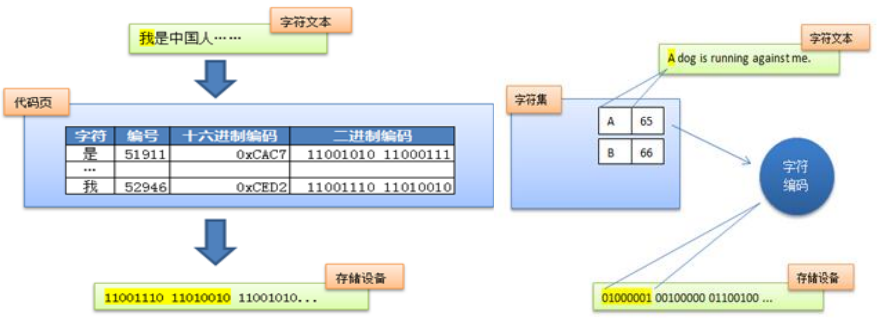
前言4：代码页是字符集编码的具体实现，你可以把他理解为一张“字符-字节”映射表，通过查表实现“字符-字节”的翻译。而对于多字节字符集，代码页中通常会有很多码表。那么程序怎么知道该使用哪张码表去解码二进制流呢？答案是，根据第一个字节来选择不同的码表进行解析。

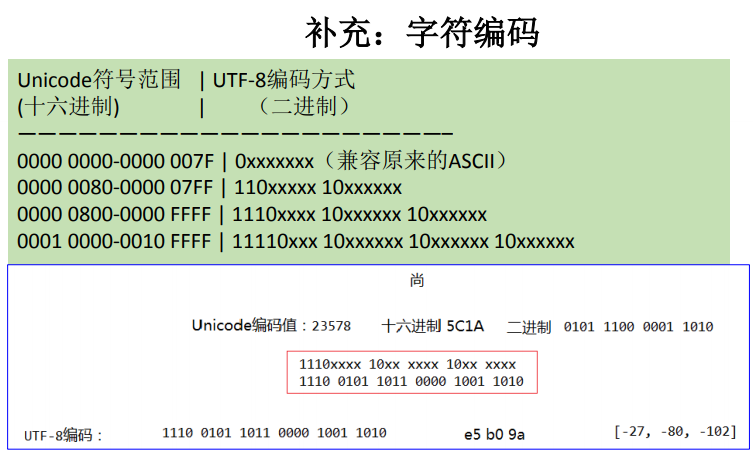
前言5：虽然通过使用不同字符集，我们可以在一台机器上查阅不同语言的文档，但是我们仍然无法解决一个问题：在一份文档中显示所有字符。为了解决这个问题，我们需要一个全人类达成共识的巨大的字符集，这就是Unicode字符集。

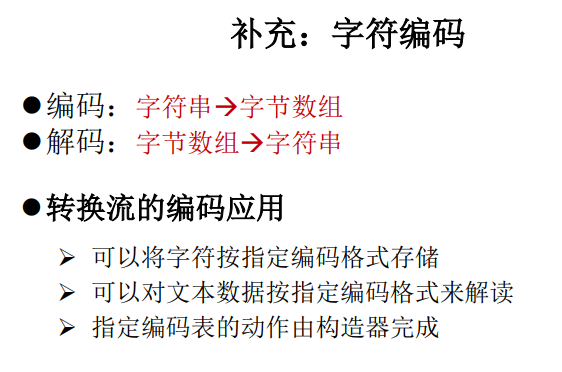
前言6：在Unicode出现之前，所有的字符集都是和具体编码方案绑定在一起的，例如，ASCII编码系统规定使用7比特来编码ASCII字符集；GB2312以及GBK字符集，限定了使用最多2个字节来编码所有字符，并且规定了字节序。Unicode在设计上考虑到了这一点，将字符集和字符编码方案分离开。也就是说，虽然每个字符在Unicode字符集中都能找到唯一确定的编号（字符码，又称Unicode码），但是决定最终字节流的却是具体的字符编码。例如同样是对Unicode字符“A”进行编码，UTF-8字符编码得到的字节流是0x41，而UTF-16（大端模式）得到的是0x00 0x41。

常见的编码表：

1. ASCII：美国标准信息交换码。
   1. 用一个字节的7位可以表示。
2. ISO8859-1：拉丁码表。欧洲码表
   1. 用一个字节的8位表示。
3. GB2312：中国的中文编码表。
   1. 最多两个字节编码所有字符
4. GBK：中国的中文编码表升级，融合了更多的中文文字符号。
   1. 最多两个字节编码，GBK等双字节编码方式，用最高位是1或0表示两个字节和一个字节。
5. Unicode：国际标准码，融合了目前人类使用的所有字符。为每个字符分配唯一的字符码。Unicode只是定义了一个庞大的、全球通用的字符集，并为每个字符规定了唯一确定的编号，具体存储成什么样的字节流，取决于字符编码方案。
   1. 所有的文字都用两个字节来表示。
6. UTF-8：变长的编码方式，
   1. 可用1-4个字节来表示一个字符。







相关问题：

1. 带签名的UTF-8指的是什么意思？

带签名指的是字节流以BOM标记开始。很多软件会“智能”的探测当前字节流使用的字符编码，这种探测过程出于效率考虑，通常会提取字节流前面若干个字节，看看是否符合某些常见字符编码的编码规则。由于UTF-8和ASCII编码对于纯英文的编码是一样的，无法区分开来，因此通过在字节流最前面添加BOM标记可以告诉软件，当前使用的是Unicode编码，判别成功率就十分准确了。但是需要注意，不是所有软件或者程序都能正确处理BOM标记，例如PHP就不会检测BOM标记，直接把它当普通字节流解析了。因此如果你的PHP文件是采用带BOM标记的UTF-8进行编码的，那么有可能会出现问题。

1. 例如最常见的网页乱码问题。如果你是网站技术人员，遇到这样的问题，需要检查以下原因：

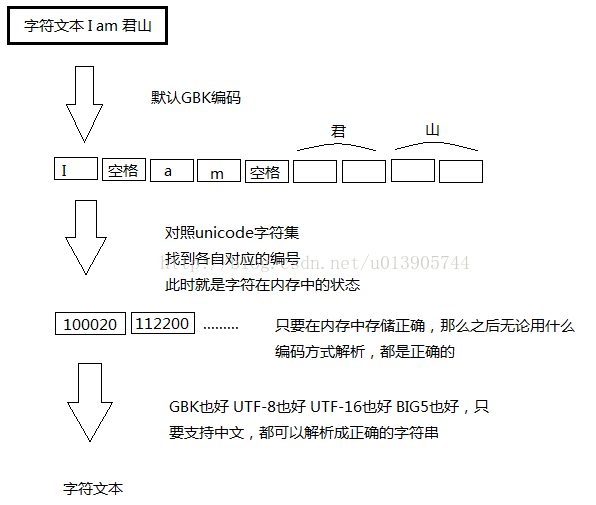
服务器返回的响应头Content-Type没有指明字符编码

网页内是否使用META HTTP-EQUIV标签指定了字符编码

网页文件本身存储时使用的字符编码和网页声明的字符编码是否一致

字符在JAVA内存的存在：

例1：读取GBK文件



工作原理

在将文本从外部文件读入内存中时，存储到内存中byte[]中。在JVM中、在内存中、在代码里声明的每一个char、String类型的变量中字符以unicode格式存在。如果你在eclipse将java文件properties的默认编码方式有GBK改为UTF-8，就会乱码。因为你使用UTF-8来解码GBK编码的字节流。具体原理分析见2.