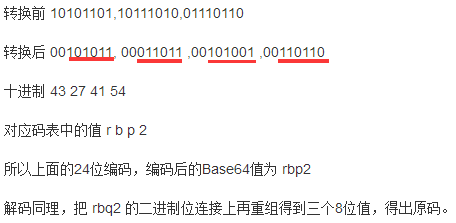
BASE64加密解密

# BASE64名称的由来

编码表的大小为**2^6=64**，这也是Base64名称的由来。

# BASE64的原理

Base64编码要求把**3个8位字节（3\*8=24）**转化为**4个6位的字节（4\*6=24）**，之后在6位的前面补两个0，形成**8位一个字节**的形式。**如果剩下的字符不足3个字节，则用0填充，输出字符使用‘=’，因此编码后输出的文本末尾可能会出现1或2个‘=’。示例如下:**



原文的字节数量应该是3的倍数，如果这个条件不能满足的话，那该怎么办呢?

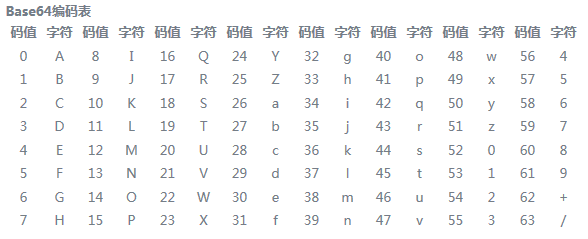
在编码过程中，对于原码是以3个字节为一组，变成4个字节。对于不足的，自动补0。如果原码字节数目n%3=0,则Base64编码中不存在=；如果n%3=1，则编码后存在两个=号，即以==结尾；如果n%3==2，则编码后存在一个=号，即以一个**=**结尾。原因如下：

### 11111111 –>00111111 00110000 00000000 00000000 🡪/w==

### 11111111 11111111🡪00111111 00111111 00111100 00000000 🡪//8=

# 标准Base64编码表

Base64编码表中只有64个。因为原来的3个字节变成4个字节，此时新的字节就是由高**位2个0和低六位**组成，所以大小范围为0-63，即2^6=64。



**注意**：码值为0与添加的新字节为0编码不相同。

**码值64个，编码字符是65个。**

如果正好码值为0而非添加的字节，则对应编码字符为A，如@编码为QA==

@ ： 01000000—>00**010000** 00**00**0000 00000000 00000000🡪QA==

附 : ASCII表可以参照<http://www.asciima.com/>

# Base64编码的不足及变种

**标准的Base64**并不适合直接放在URL里传输，因为URL编码器会把标准Base64中的“/”和“+”字符变为形如“%XX”的形式，而这些“%”号在存入数据库时还需要再进行转换，因为ANSI SQL中已将“%”号用作通配符。

为解决此问题，可采用一种用于URL的改进Base64编码，它不仅在末尾填充'='号，并将标准Base64中的“+”和“/”分别改成了“-”和“\_”，这样就免去了在URL编解码和数据库存储时所要作的转换，避免了编码信息长度在此过程中的增加，并统一了数据库、表单等处对象标识符的格式。

另有一种用于正则表达式的改进Base64变种，它将“+”和“/”改成了“!”和“-”，因为“+”,“\*”以及前面在IRCu中用到的“[”和“]”在正则表达式中都可能具有特殊含义。

此外还有一些变种，它们将“+/”改为“\_-”或“.\_”（用作编程语言中的标识符名称）或“.-”（用于XML中的Nmtoken）甚至“\_:”（用于XML中的Name）。

# Base64的应用

Base64编码可用于在HTTP环境下传递较长的标识信息。

先以“迅雷下载”为例： 很多下载类网站都提供“迅雷下载”的链接，其地址通常是加密的迅雷专用下载地址。

其实迅雷的“专用地址”也是用Base64"加密"的，其过程如下：

一、在地址的前后分别添加AA和ZZ

二、对新的字符串进行Base64编码

另：　Flashget的与迅雷类似，只不过在第一步时加的“料”不同罢了，Flashget在地址前后加的“料”是[FLASHGET]；而QQ旋风的干脆不加料，直接就对地址进行Base64编码了。

Mozilla Thunderbird和Evolution用Base64来保密电子邮件密码

Base64 也会经常用作一个简单的“加密”来保护某些数据，而真正的加密通常都比较繁琐。垃圾讯息传播者用Base64来避过反垃圾邮件工具，因为那些工具通常都不会翻译Base64的讯息。在LDIF档案，Base64用作编码字串。

# Base64编码解码代码实现

<http://baike.baidu.com/link?url=OceDk5cztQeCZ3n5Jyi86_xs1zIddfX4mO36Fdfvq3nL_GcQKMtqBfHm5M00kxmD-aZWGSB8FiiF4CS15uqjNq>

Java、Js、C#、PHP等。

## 编码思路

### \* Base64编码实现原理：

\* 1. 传进去的参数是待编码的字符串，首先转换成字节数组（这里有个字符集的问题）

\* 2. 首先处理前3的最大倍数的字节，每3个一组，这里涉及到一些小方法，

\* 需要3\*8==》4\*8，生成4个字节为线，生成一个字节用单独一个方法，

\* 则需要四个小方法，返回的都是相应的字节：

\* 2.1 firstByte(byte b) 参数只需要3个字节中的第一个字节即可；

\* 2.2 secondByte(byte last\_b,byte next\_b) 参数需要3个字节中的第1个字节和第2个字节；

\* 2.3 thirdByte(byte last\_b,byte next\_b) 参数需要3个字节中的第2个字节和第3个字节；

\* 2.4 fourthByte(byte b) 参数需要3个字节中的第3个字节。

\* 另外，对于最后可能会留下1个字节或2个字节不能组成一组，需要单独处理：

\* 但是剩下1个字节时，生成第一个字节仍然调用firstByte(byte)方法，

\* 只需处理生成第二个字节， 第3、4个直接拼接=即可。

\* 剩下2个字节时，生成第1、2个字节用firstByte(byte b)、secondByte(byte last\_b,byte

\* next\_b)即可， 但是 第3个字节需要单独处理，第4个直接拼接=即可。

\* 对于上面的单独处理，有共性就是取出特定原始字节的某些位到一个字节的00后上，

\* 不够的用0补齐，这个方法 就是lastOneByte(byte b,int move)。

\* 3. 把转换后的字节（0-63）当成编码表的索引，从而查出编码所需要的字符的字节，利用new String(byte[])即转换成字符串，利用StringBuilder或StringBuffer拼接即可。需要利用方法baseIndex(byte b)。

### Base解码原理（与编码类似）：

* 1. 首先把待解码的字符串最后的=号去除。正常情况下，输入的待解码的字符串的个数及字节数目必须是4的整数倍。对于特殊的异常情况，可以直接抛出异常处理。正常情况下，后面跟1个或2个=。
  2. 同样先把前面是4的倍数的字节转换成相应字节，然后利用new String进行转换成字符串，利用StringBuilder进行拼接。

同理需要3个小方法，用来4\*8🡺3\*8。first、seconde、third。

不需要使用last。

注意：编码和解码统一使用UTF-8,byte[].getBytes()和new String时需要指定。

## Java实现代码：

见txt文件。