Base64编码解码的原理及应用

# Base64名称的由来

由于Base64的编码表的大小为26=64个，且是由26个大写字母、26个小写字母和10个数字以及“+”、“/”最基本的符号组成，故称之为Base64。

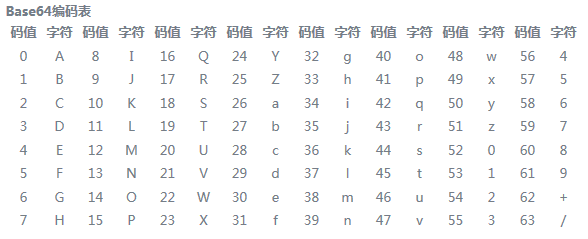
(**有时间就好好看看源码，怎么实现的。**)

# Base64编码解码的原理

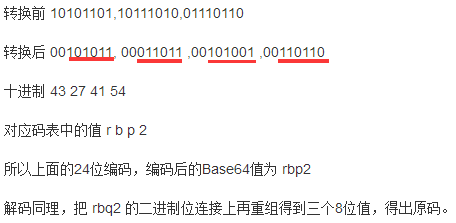
**最简单的记忆：3个字节编程4个字节。3\*8=4\*6，4个字节的高2位都补0。**

Base64的Alphabet表如下表所示：





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=

# 注意

**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==

因为1个字节8位，那么后面需要加两个==，而第一个、第二个字节都需要按照Base64编码表编码，其中00010000为Q，00000000为A。而后面的00000000需要自动编码为=。故称为64个码值，对应65个编码符号；多的那个符号为=。

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

# Base64编码的不足以及改进

## URL中的问题

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。

改进：将“+”换成“-”；将“/”换成“\_”。

示例：

Base64.Encoder base64Encoder = **Base64.getEncoder();//**最基本的Base64

Base64.Decoder base64Decoder = **Base64.getDecoder();**

String str = "中国是一个伟大的国家！山东省菏泽曹县是一个大县城。";

helper(base64Encoder,base64Decoder,str);

base64Encoder = **Base64.getUrlEncoder();**

base64Decoder = **Base64.getUrlDecoder();**

helper(base64Encoder,base64Decoder,str);//仅仅只是将+变成-，将/变成\_

最基本的：1tC5**+**srH0ru49s6wtPO1xLn6vNKjocm9tqvKobrK1POy3M**/**YysfSu7j2tPPP2LPHoaM=

Url编码：

1tC5**-**srH0ru49s6wtPO1xLn6vNKjocm9tqvKobrK1POy3M**\_**YysfSu7j2tPPP2LPHoaM=

## 正则表达式中的问题

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）。

# 标准Base64与UrlBase64、MimeBase64的区别

## 标准Base64与UrlBase64的区别

将“+”换成“-”；将“/”换成“\_”。

## 标准Base64与MimeBase64的区别

MimeBase64是76个字节自动换行。

The encoded output must be represented in lines of no more than 76 characters each and uses a carriage **return '\r' followed immediately by a linefeed '\n' as the line separator**. No line separator is added to the end of the encoded output. All line separators or other characters not found in the base64 alphabet table are ignored in decoding operation.

Base64.Encoder base64Encoder = Base64.*getEncoder*();

Base64.Decoder base64Decoder = Base64.*getDecoder*();

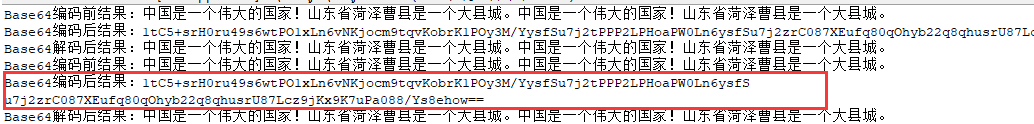
String str = "中国是一个伟大的国家！山东省菏泽曹县是一个大县城。中国是一个伟大的国家！山东省菏泽曹县是一个大县城。";

*helper*(base64Encoder,base64Decoder,str);

base64Encoder = Base64.***getMimeEncoder***();

base64Decoder = Base64.***getMimeDecoder***();

*helper*(base64Encoder,base64Decoder,str);



public static void helper(Base64.Encoder base64Encoder,Base64.Decoder base64Decoder,String str){

String result = base64Encoder.encodeToString(str.getBytes());

System.out.println("Base64编码前结果："+str);

System.out.println("Base64编码后结果："+result);

String original = new String(base64Decoder.decode(result));

System.out.println("Base64解码后结果："+original);

}

# 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

## JDK中的实现

JDK1.8已经提供了Base64的编解码方案。可以直接看源代码。

java.util.Base64以及内部类Base64.Encoder和Base64.Decoder。

目前提供了三对编码器和解码器：都是通过Base64的静态方法获取，分别为

标准的：Base64.getEncoder()和Base64.getDecoder()

Url的：Base64.getUrlEncoder()和Base64.getUrlDecoder()

MIME：Base64.getMimeEncoder()和Base64.getMimeDecoder()

源代码：

private int encode0(byte[] src, int off, int end, byte[] dst) {

char[] base64 = isURL ? toBase64URL : toBase64;

int sp = off;

int slen = (end - off) / 3 \* 3;

int sl = off + slen;

if (linemax > 0 && slen > linemax / 4 \* 3)

slen = linemax / 4 \* 3;

int dp = 0;

while (sp < sl) {

int sl0 = Math.min(sp + slen, sl);

for (int sp0 = sp, dp0 = dp ; sp0 < sl0; ) {

int bits = (src[sp0++] & 0xff) << 16 |

(src[sp0++] & 0xff) << 8 |

(src[sp0++] & 0xff);

dst[dp0++] = (byte)base64[(bits >>> 18) & 0x3f];

dst[dp0++] = (byte)base64[(bits >>> 12) & 0x3f];

dst[dp0++] = (byte)base64[(bits >>> 6) & 0x3f];

dst[dp0++] = (byte)base64[bits & 0x3f];

}

int dlen = (sl0 - sp) / 3 \* 4;

dp += dlen;

sp = sl0;

if (dlen == linemax && sp < end) {

for (byte b : newline){

dst[dp++] = b;

}

}

}

if (sp < end) { // 1 or 2 leftover bytes

int b0 = src[sp++] & 0xff;

dst[dp++] = (byte)base64[b0 >> 2];

if (sp == end) {

dst[dp++] = (byte)base64[(b0 << 4) & 0x3f];

if (doPadding) {

dst[dp++] = '=';

dst[dp++] = '=';

}

} else {

int b1 = src[sp++] & 0xff;

dst[dp++] = (byte)base64[(b0 << 4) & 0x3f | (b1 >> 4)];

dst[dp++] = (byte)base64[(b1 << 2) & 0x3f];

if (doPadding) {

dst[dp++] = '=';

}

}

}

return dp;

}

}

## 网络学习

http://baike.baidu.com/link?url=OceDk5cztQeCZ3n5Jyi86\_xs1zIddfX4mO36Fdfvq3nL\_GcQKMtqBfHm5M00kxmD-aZWGSB8FiiF4CS15uqjNq

Java、Js、C#、PHP等。

1) 编码思路

A. \* 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)。

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。

2) Java实现代码：

见txt文件。