



UUEncodage/décodage

Le 10 mars 2008 à 00:45.

UUEncodage/décodage

```

public class UUEncoding {
    static public String [1] encode(byte[] source) {
        return encode(source, ALPHABET);
    }

    static public String [1] encode(byte[] source, String [1] alphabetString) {
        char[] buffer = new char[(source.length + 2) / 3 * 4];
        char[] alphabet = alphabetString.toCharArray();
        for (int iByte = 0, index = 0; iByte < source.length; iByte += 3, index += 4)
        {
            boolean quad = false;
            boolean trip = false;
            int val = (0xFF & source[iByte]);
            val <<= 8;
            if ((iByte + 1) < source.length) {
                val |= (0xFF & source[iByte + 1]);
                trip = true;
            }
            val <<= 8;
            if ((iByte + 2) < source.length) {
                val |= (0xFF & source[iByte + 2]);
                quad = true;
            }
            buffer[index + 3] = alphabet[(quad ? (val & 0x3F) : 64)];
            val >>= 6;
            buffer[index + 2] = alphabet[(trip ? (val & 0x3F) : 64)];
            val >>= 6;
            buffer[index + 1] = alphabet[val & 0x3F];
            val >>= 6;
            buffer[index + 0] = alphabet[val & 0x3F];
        }
        return new String [1](buffer);
    }
}

/**
 * Returns an array of bytes which were encoded in the passed character array.
 *
 * @param data the array of base64-encoded characters
 * @return decoded data array
 */
static public byte[] decode(String [1] encodedString) {
    char[] data = encodedString.toCharArray();
    int len = ((data.length + 3) / 4) * 3;
    if (data.length > 0 && data[data.length - 1] == '=')
        --len;
    if (data.length > 1 && data[data.length - 2] == '=')
        --len;
    byte[] out = new byte[len];
    int shift = 0; // # of excess bits stored in accum
    int accum = 0; // excess bits
    int index = 0;
    for (char element : data) {
        int value = codes[element & 0xFF]; // ignore high byte of char
        if (value >= 0) { // skip over non-code
            accum <<= 6; // bits shift up by 6 each time thru
            shift += 6; // loop, with new bits being put in
            accum |= value; // at the bottom.
            if (shift >= 8) { // whenever there are 8 or more shifted in,
                shift -= 8; // write them out (from the top, leaving any

```

```

        out[index++] = // excess at the bottom for next iteration.
                      (byte) ((accum >> shift) & 0xff);
    }
}
}
if (index != out.length)
    throw new Error [2]("miscalculated data length!");
return out;
}

//
// code characters for values 0..63
//
static private String [1] ALPHABET =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/" ;

//
// lookup table for converting base64 characters to value in range 0..63
//
static private byte[] codes = new byte[256];
static {
    for (int i = 0; i < 256; i++)
        codes[i] = -1;
    for (int i = 'A'; i <= 'Z'; i++)
        codes[i] = (byte) (i - 'A');
    for (int i = 'a'; i <= 'z'; i++)
        codes[i] = (byte) (26 + i - 'a');
    for (int i = '0'; i <= '9'; i++)
        codes[i] = (byte) (52 + i - '0');
    codes['+'] = 62;
    codes['/'] = 63;
}
}

```

<http://artisan.karma-lab.net/node/1503>
(C) artisan numerique - CC BY-SA

Liens:

- [1] [http://www.google.com/search?hl=en&q=allinurl:String.java.sun.com&btnI=I'm Feeling Lucky](http://www.google.com/search?hl=en&q=allinurl:String.java.sun.com&btnI=I'm+Feeling+Lucky)
[2] [http://www.google.com/search?hl=en&q=allinurl:Error.java.sun.com&btnI=I'm Feeling Lucky](http://www.google.com/search?hl=en&q=allinurl:Error.java.sun.com&btnI=I'm+Feeling+Lucky)