LGTM 1063

711

@PlanetMeron

投稿日 2015年10月14日　更新日 2018年04月13日

base64ってなんぞ？？理解のために実装してみた

base64

base64とは

base64という言葉を聞いたことがあるでしょうか？？
base64とは、64進数を意味する言葉で、すべてのデータをアルファベット(`a~z`, `A~z`)と数字(`0~9`)、一部の記号(`+`, `/`)の**64文字**で表すエンコード方式です
ただ、データ長を揃えるためにパディングとして末尾に記号の`=`を使用するので、厳密にはbase64は、65文字の英数字から表現されます
(URLや正規表現のなかでbase64を用いると一部の記号(`+`, `/`)は特別な意味を持つことがあるので`-`や`_`などが用いられることがあります)

なぜbase64をつかうの？？

かつての電子メールを送るためのプロトコルSMTPでは、ASCIIといわれる7bitで表現される英数字しか送ることができませんでした
したがって、メールを使って画像や音声などのデータをやりとりしたいと思った時に、英数字しか対応していないSMTPでは、それらのデータを送受信することができませんでした

そこで、すべてのデータを英数字で表すMIME(Multipurpose Internet Mail Extensions)という規格が登場し、その中でbase64というデータの変換方法が定められました
これによって、受信側と送信側がMIMEに則ってエンコード・デコードをすることで、メールを通して画像や音声などの送受信が可能になりました

現在では、JSONなどで特殊文字を含まないように画像データをbase64でエンコードしたり、Webページの表示の際にリクエスト数を減らすためにbase64でエンコードした画像をhtmlにそのまま埋め込むなどの用途で用いられています

base64実装してみた

どの言語にもbase64はサポートされていて、base64の実装をする必要はほとんどありませんが、エンコードの原理の理解を含めて実装してみようと思います

base64の変換方式

base64における処理を簡潔にいうと、
エンコードしたいファイルのバイナリデータを6bitずつ取り出し(足りない分は0を追加する)、
6ビットとAscii文字の変換表を用いて、4文字ずつにする(4文字に満たない場合は`=`を追加する)

変換アルゴリズム

ここでは、文字列をバイナリ化し、base64でエンコードする例を通して、具体的な処理手順について確認をします

処理手順を以下に示します

1. 変更したい文字列をバイナリ(２進数)に変換する

```
変換したい文字列"abcdefg"
→0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67(16進数)
→0110 0001, 0110 0010, 0110 0011, 0110 0100, 0110  0101, 0110 0110, 0110 0111(2進数)
```

2. バイナリを6ビットづつに分割

```
011000, 010110, 001001, 100011, 011001, 000110, 0101 01, 100110, 011001,11
```

3. 最後の2ビットが余るので,6ビットになるように0を追加する

```
011000, 010110, 001001, 100011, 011001, 000110, 0101 01, 100110, 011001, 110000
```

4. 下図に示す変換表よりビットを文字に変換する

```
Y W J j Z G V m Z w
```

5. 4文字に分けた時に、2文字分足りないので`=`を追加する

```
YWJj ZGVm Zw==
```

6. base64の文字列の完成

```
"YWJjZGVmZw=="
```

~base64のビット列と英数字の変換表~

10進	2進	文字		10進	2進	文字		10進	2進	文字		10進	2進	文字
0	000000	A		16	010000	Q		32	100000	g		48	110000	w
1	000001	B		17	010001	R		33	100001	h		49	110001	x
2	000010	C		18	010010	S		34	100010	i		50	110010	y
3	000011	D		19	010011	T		35	100011	j		51	110011	z
4	000100	E		20	010100	U		36	100100	k		52	110100	0
5	000101	F		21	010101	V		37	100101	l		53	110101	1
6	000110	G		22	010110	W		38	100110	m		54	110110	2
7	000111	H		23	010111	X		39	100111	n		55	110111	3
8	001000	I		24	011000	Y		40	101000	o		56	111000	4
9	001001	J		25	011001	Z		41	101001	p		57	111001	5
10	001010	K		26	011010	a		42	101010	q		58	111010	6
11	001011	L		27	011011	b		43	101011	r		59	111011	7
12	001100	M		28	011100	c		44	101100	s		60	111100	8
13	001101	N		29	011101	d		45	101101	t		61	111101	9
14	001110	O		30	011110	e		46	101110	u		62	111110	+
15	001111	P		31	011111	f		47	101111	v		63	111111	/

プログラムで作ってみた

以下のプログラムの実行には、変換表(base64_table.json)が必要なので別途、ダウンロードして使ってください

string2base64.py

```python
#!/usr/bin/env python

import sys
import json

# 1文字が8バイトで表される
BYTE_SIZE = 8

# 文字列textをバイナリの文字列に変換する
def str2bin(s):
    binStr = ""

    #ord関数で文字をASCIIコードへ変換し、
    #シフト演算で各桁のビットを取り出す
    for c in s:
        for i in range(BYTE_SIZE):
            binStr += str((ord(c) >> (BYTE_SIZE - (i + 1))) & 1)
    return binStr

# 文字列sを文字数nで分割したリストを返す
# (例)
# split("abcdef", 2) => [["ab"], ["cd"], ["ef"]]
def split(s, n):
    return [s[i:i+n] for i in range(0, len(s), n)]



# 文字列sを文字数nで分割した時に足りない部分を文字cで埋める
# (例)
# fillInBlank("abcd", 6, "=") => "abcd=="
def fillInBlank(s, n, c):
    mod = len(s) % n

    # 割り切れた場合は何も処理をしない
    if mod == 0: return s

    # 割り切れなかった場合、残りの部分を埋める
    margin = n - mod
    return s + c * margin

# main関数
def main():

    # コマンドの引数を受け取る
    argvs = sys.argv
    argc = len(argvs)

    # 引数が1つじゃなかったら無かったら、処理をせず終了
    if argc != 2:
        print "Usage:\n$ python %s CONVERT_STRING" % argvs[0]
        quit()

    # 1. 文字列をバイナリ文字列に変換
    binStr = str2bin(argvs[1])

    # 2. バイナリを6ビットづつに分割
    splitCount  =6
    s = split(binStr, splitCount)

    # 最後の2ビットが余るので,6ビットになるように0を追加する
    s[-1] = fillInBlank(s[-1], 6, "0")

    # 変換表の辞書を読み込み
    tableFile = open("base64_table.json", "r")
    base64Dict = json.loads(tableFile.read())

    result = ""

    for i in s:
        result += base64Dict[i]

    print result

    print fillInBlank(result, 4, "=")

if __name__ == '__main__':
    main()
```

ファイルの実行

文字列"abcdefg"を引数としてpythonファイルを実行すると次のような結果がでます

Terminal

```
$ python string2base64.py abcdefg
binary:  0110000101100010011000110110010001100101011001100110 0111
base64:  YWJjZGVmZw==
```

WEB便利ツールさんのように、外部ツールと同じ結果を得ることができたので、成功です！！

まとめ

文字列をBase64でエンコードするのは、そもそもの用途からは離れてしまうのですが、確認のために実装を行いました。

車輪の再発明ですが、理解ができて、思った通りに結果になるのは楽しいですね！！

参考文献
https://ja.wikipedia.org/wiki/Base64
http://www.sophia-it.com/content/BASE64

Ishibashi Genki
@PlanetMeron
フォロー