链滴

Java C# LZMA 字符串压缩解压

作者：88250

原文链接：https://ld246.com/article/1481701879422

来源网站：链滴

压缩：将指定的原字符串用 LZMA 算法压缩，然后以 BASE64 编码

解压：将指定的 BASE64 编码的字符串用 LZMA 解压，返回原字符串

原字符串为 UTF-8 编码。

Java 版本

导入包

基本都是 JDK 内置的包，BASE64 部分可能需要替换一下（JDK8 已经自带 BASE64）。

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.IOUtils;
```

实现部分

```
/**
 * 使用 lzma 进行压缩.
 *
 * @param str 压缩前的文本
 * @return 压缩后的文本（BASE64 编码）
 * @throws IOException 如果解压异常
 */
public static String lzma(final String str) throws IOException {
    if (str == null || "".equals(str)) {
        return str;
    }

    final SevenZip.Compression.LZMA.Encoder encoder = new SevenZip.Compression.LZMA.En
oder();

    final ByteArrayOutputStream out = new ByteArrayOutputStream();
    final ByteArrayInputStream in = new ByteArrayInputStream(str.getBytes());

    try {
        encoder.SetEndMarkerMode(false);
        encoder.WriteCoderProperties(out);
        final long fileSize = in.available();
        for (int i = 0; i < 8; i++) {
            out.write((int) (fileSize >>> (8 * i)) & 0xFF);
        }
        encoder.Code(in, out, -1, -1, null);

        final byte[] compressed = out.toByteArray();

        return new String(Base64.encodeBase64(compressed), "UTF-8");
    } finally {
        IOUtils.closeQuietly(in);
```

```
        IOUtils.closeQuietly(out);
    }
}

/**
 * 使用 lzma 进行解压缩.
 *
 * @param compressedStr 压缩后的文本（BASE64 编码）
 * @return 解压后的文本
 * @throws IOException 如果解压异常
 */
public static String unlzma(final String compressedStr) throws IOException {
    if (null == compressedStr || "".equals(compressedStr)) {
        return compressedStr;
    }

    final SevenZip.Compression.LZMA.Decoder decoder = new SevenZip.Compression.LZMA.D
coder();

    final ByteArrayOutputStream out = new ByteArrayOutputStream();
    final ByteArrayInputStream in = new ByteArrayInputStream(Base64.decodeBase64(compres
edStr));

    try {
        final int propertiesSize = 5;
        final byte[] properties = new byte[propertiesSize];
        if (in.read(properties, 0, propertiesSize) != propertiesSize) {
            throw new IOException("input .lzma file is too short");
        }
        if (!decoder.SetDecoderProperties(properties)) {
            throw new IOException("Incorrect stream properties");
        }
        long outSize = 0;
        for (int i = 0; i < 8; i++) {
            final int v = in.read();
            if (v < 0) {
                throw new IOException("Can't read stream size");
            }
            outSize |= ((long) v) << (8 * i);
        }
        if (!decoder.Code(in, out, outSize)) {
            throw new IOException("Error in data stream");
        }

        return out.toString("UTF-8");
    } finally {
        IOUtils.closeQuietly(in);
        IOUtils.closeQuietly(out);
    }
}
```

C# 版本

命名空间

```
using System;
using System.Text;
using System.IO;
using SevenZip;
```

实现部分

```
/// <summary>
/// 使用 lzma 进行压缩
/// </summary>
/// <param name="str">压缩前的文本</param>
/// <returns>压缩后的文本（BASE64 编码）</returns>
public static string lzma(string str)
{
    if (null == str || "".Equals(str))
    {
        return str;
    }

    byte[] buffer = Encoding.UTF8.GetBytes(str);
    byte[] compressed = SevenZipHelper.Compress(buffer);

    return Convert.ToBase64String(compressed);
}

/// <summary>
/// 使用 lzma 进行解压缩
/// </summary>
/// <param name="compressedStr">压缩后的文本（BASE64 编码）</param>
/// <returns>解压后的文本</returns>
public static string unlzma(string compressedStr)
{
    if (null == compressedStr || "".Equals(compressedStr))
    {
        return compressedStr;
    }

    byte[] decompressed = SevenZipHelper.Decompress(Convert.FromBase64String(compress
dStr));


    return Encoding.UTF8.GetString(decompressed);
}


public static class SevenZipHelper
{

    static int dictionary = 1 << 23;

    // static Int32 posStateBits = 2;
```

```
    // static  Int32 litContextBits = 3; // for normal files
    // UInt32 litContextBits = 0; // for 32-bit data
    // static  Int32 litPosBits = 0;
    // UInt32 litPosBits = 2; // for 32-bit data
    // static   Int32 algorithm = 2;
    // static    Int32 numFastBytes = 128;
    static bool eos = false;

    static CoderPropID[] propIDs =
            {
            CoderPropID.DictionarySize,
            CoderPropID.PosStateBits,
            CoderPropID.LitContextBits,
            CoderPropID.LitPosBits,
            CoderPropID.Algorithm,
            CoderPropID.NumFastBytes,
            CoderPropID.MatchFinder,
            CoderPropID.EndMarker
        };

    // these are the default properties, keeping it simple for now:
    static object[] properties =
            {
            (Int32)(dictionary),
            (Int32)(2),
            (Int32)(3),
            (Int32)(0),
            (Int32)(2),
            (Int32)(128),
            "bt4",
            eos
        };

    public static byte[] Compress(byte[] inputBytes)
    {
        MemoryStream inStream = new MemoryStream(inputBytes);
        MemoryStream outStream = new MemoryStream();
        SevenZip.Compression.LZMA.Encoder encoder = new SevenZip.Compression.LZMA.Enco
er();
        encoder.SetCoderProperties(propIDs, properties);
        encoder.WriteCoderProperties(outStream);
        long fileSize = inStream.Length;
        for (int i = 0; i < 8; i++)
            outStream.WriteByte((Byte)(fileSize >> (8 * i)));
        encoder.Code(inStream, outStream, -1, -1, null);

        return outStream.ToArray();
    }

    public static byte[] Decompress(byte[] inputBytes)
    {
        MemoryStream newInStream = new MemoryStream(inputBytes);

        SevenZip.Compression.LZMA.Decoder decoder = new SevenZip.Compression.LZMA.Dec
```

```
der();

        newInStream.Seek(0, 0);
        MemoryStream newOutStream = new MemoryStream();

        byte[] properties2 = new byte[5];
        if (newInStream.Read(properties2, 0, 5) != 5)
            throw (new Exception("input .lzma is too short"));
        long outSize = 0;
        for (int i = 0; i < 8; i++)
        {
            int v = newInStream.ReadByte();
            if (v < 0)
                throw (new Exception("Can't Read 1"));
            outSize |= ((long)(byte)v) << (8 * i);
        }
        decoder.SetDecoderProperties(properties2);

        long compressedSize = newInStream.Length - newInStream.Position;
        decoder.Code(newInStream, newOutStream, compressedSize, outSize, null);

        byte[] b = newOutStream.ToArray();

        return b;
    }
}
```

姊妹篇： https://hacpai.com/article/1480569906672