CodeJava
Coding Your Passion

Search...

JVMHOST
All kinds of Java hosting since 2010
FREE 30-DAY TRIAL
INSTANT ACTIVATION
PRIVATE JVM + cPanel
BEST PRICE GUARANTEE

Featured Books

- Java Coding Guideli Recommendations f Reliable and Secure Programs (SEI Series Software Engineerin
- Head First Design Pa
- Java Concurrency ir
- Java Performance
- Java Puzzlers: Traps and Corner Cases
- Head First Object-O

Analysis and Design

Clean Code: A Hand
Agile Software Craft

Algorithms Unlocke

Data Structures and
Algorithm Analysis i
(3rd Edition)

Refactoring: Improvi
Design of Existing C

The Pragmatic Progr
From Journeyman to

Code Complete: A P
Handbook of Softwa
Construction, Secor

Cracking the Coding
Interview: 150 Progr
Questions and Solut

The Clean Coder: A
Conduct for Profess
Programmers (Rober
Martin Series)

How to calculate MD5 and SHA hash values in Java

Last Updated on 06 September 2014 | Print Email

Java Performance **Clean Code** **The Clean Coder** **The Pragmatic Programmer**

In cryptography, **MD5** (*Message Digest version 5*) and **SHA** (*Secure Hash Algorithm*) are two well-known message digest algorithms. They are also referred as cryptographic hash functions, which take arbitrary-sized data as input (message) and produce a fixed-length hash value. One of the most important properties of hash functions is, it's infeasible to generate a message that has a given hash (secure one-way). Hash functions are frequently used to check data integrity such as checking integrity of a downloaded file against its publicly-known hash value. Another common usage is to encrypt user's password in database.

The Java platform provides two implementation of hashing functions: MD5 (produces 128-bit hash value), SHA-1 (160-bit) and SHA-2 (256-bit). This tutorial demonstrates how to generate MD5 and SHA hash values from String or file using Java.

Here are general steps to generate a hash value from an input (message):

- First approach (suitable for small-sized message):

```
// algorithm can be "MD5", "SHA-1", "SHA-256"
MessageDigest digest = MessageDigest.getInstance(algorithm);

byte[] inputBytes = // get bytes array from message

byte[] hashBytes = digest.digest(inputBytes);

// convert hash bytes to string (usually in hexadecimal form)
```

- Second approach (suitable for large-size message, i.e. large file):

```
MessageDigest digest = MessageDigest.getInstance(algorithm);

byte[] inputBytes = // get bytes array from message

digest.update(inputBytes);

byte[] hashedBytes = digest.digest();

// convert hash bytes to string (usually in hexadecimal form)
```

Now, let’s see some examples in details.

1. Generating Hash from String

The following method takes a message and algorithm name as inputs and returns hexadecimal form of the calculated hash value:

```
private static String hashString(String message, String algorithm
        throws HashGenerationException {

    try {
        MessageDigest digest = MessageDigest.getInstance(algorith
        byte[] hashedBytes = digest.digest(message.getBytes("UTF-

        return convertByteArrayToHexString(hashedBytes);
    } catch (NoSuchAlgorithmException | UnsupportedEncodingExcept
        throw new HashGenerationException(
                "Could not generate hash from String", ex);
    }
}
```

The `HashGenerationException` is a custom exception (you can find this class in the attachment). The `convertByteArrayToHexString()` method is implemented as follows:

```
private static String convertByteArrayToHexString(byte[] arrayByte
    StringBuffer stringBuffer = new StringBuffer();
    for (int i = 0; i < arrayBytes.length; i++) {
        stringBuffer.append(Integer.toString((arrayBytes[i] & 0xff
                .substring(1));
    }
    return stringBuffer.toString();
}
```

The **`hashString()`** is a general method. Here are four public utility methods that are specific to each algorithm (MD5, SHA-1 and SHA-256):

```
public static String generateMD5(String message) throws HashGener
    return hashString(message, "MD5");
}

public static String generateSHA1(String message) throws HashGene
    return hashString(message, "SHA-1");
}

public static String generateSHA256(String message) throws HashGe
    return hashString(message, "SHA-256");
}
```

Hence we have the following utility class:

```java
package net.codejava.security;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Hash functions utility class.
 * @author www.codejava.net
 *
 */
public class HashGeneratorUtils {
    private HashGeneratorUtils() {

    }

    public static String generateMD5(String message) throws HashG
        return hashString(message, "MD5");
    }

    public static String generateSHA1(String message) throws Hash
        return hashString(message, "SHA-1");
    }

    public static String generateSHA256(String message) throws Ha
        return hashString(message, "SHA-256");
    }

    private static String hashString(String message, String algor
            throws HashGenerationException {

        try {
            MessageDigest digest = MessageDigest.getInstance(algo
            byte[] hashedBytes = digest.digest(message.getBytes("

            return convertByteArrayToHexString(hashedBytes);
        } catch (NoSuchAlgorithmException | UnsupportedEncodingEx
            throw new HashGenerationException(
                    "Could not generate hash from String", ex);
        }
    }

    private static String convertByteArrayToHexString(byte[] arra
        StringBuffer stringBuffer = new StringBuffer();
        for (int i = 0; i < arrayBytes.length; i++) {
            stringBuffer.append(Integer.toString((arrayBytes[i] &
                    .substring(1));
        }
        return stringBuffer.toString();
    }
}
```

Here’s a test program:

```java
package net.codejava.security;

/**
 * Test generating hash values from String.
 * @author www.codejava.net
 *
 */
public class StringHashGeneratorExample {

    public static void main(String[] args) {
        try {
            String inputString = args[0];
            System.out.println("Input String: " + inputString);

            String md5Hash = HashGeneratorUtils.generateMD5(input
            System.out.println("MD5 Hash: " + md5Hash);

            String sha1Hash = HashGeneratorUtils.generateSHA1(inp
            System.out.println("SHA-1 Hash: " + sha1Hash);

            String sha256Hash = HashGeneratorUtils.generateSHA256
            System.out.println("SHA-256 Hash: " + sha256Hash);
        } catch (HashGenerationException ex) {
            ex.printStackTrace();
        }
    }

}
```

If the input message is “admin” the test program produces the following output:

```
Input String: admin
MD5 Hash: 21232f297a57a5a743894a0e4a801fc3
SHA-1 Hash: d033e22ae348aeb5660fc2140aec35850c4da997
SHA-256 Hash: 8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f
```

2. Generating Hash from File

To calculate hash value of a large file effectively, it’s recommended to repeatedly put a chunk of bytes to the message digest, until reaching end of file. Here’s such method:

```java
private static String hashFile(File file, String algorithm)
        throws HashGenerationException {
    try (FileInputStream inputStream = new FileInputStream(file))
        MessageDigest digest = MessageDigest.getInstance(algorith

        byte[] bytesBuffer = new byte[1024];
        int bytesRead = -1;

        while ((bytesRead = inputStream.read(bytesBuffer)) != -1)
            digest.update(bytesBuffer, 0, bytesRead);
        }

        byte[] hashedBytes = digest.digest();

        return convertByteArrayToHexString(hashedBytes);
    } catch (NoSuchAlgorithmException | IOException ex) {
        throw new HashGenerationException(
                "Could not generate hash from file", ex);
    }
}
```

Here are four public methods that are specific to each algorithm:

```java
public static String generateMD5(File file) throws HashGeneration
    return hashFile(file, "MD5");
}

public static String generateSHA1(File file) throws HashGeneratic
    return hashFile(file, "SHA-1");
}

public static String generateSHA256(File file) throws HashGenerat
    return hashFile(file, "SHA-256");
}
```

And here’s a test program:

```java
package net.codejava.security;

import java.io.File;

/**
 * Test generating hash values from File.
 * @author www.codejava.net
 *
 */
public class FileHashGeneratorExample {

    public static void main(String[] args) {
        try {
            String filePath = args[0];
            System.out.println("File Path: " + filePath);
            File file = new File(filePath);

            String md5Hash = HashGeneratorUtils.generateMD5(file)
            System.out.println("MD5 Hash: " + md5Hash);

            String sha1Hash = HashGeneratorUtils.generateSHA1(fil
            System.out.println("SHA-1 Hash: " + sha1Hash);

            String sha256Hash = HashGeneratorUtils.generateSHA256
            System.out.println("SHA-256 Hash: " + sha256Hash);

        } catch (HashGenerationException ex) {
            ex.printStackTrace();
        }
    }

}
```

Example output:

```
File Path: D:\Java\PDFViewer\JPedalPDFViewer.zip
MD5 Hash: 56a86f56a18b73353e5f0afa7b142ed1
SHA-1 Hash: dc55bd7e84c4787242499ec068fa145bcca01937
SHA-256 Hash: 093059d79d009662a0a7f70c74cec934a73c1becc8ac813cdcc4
```

NOTES:

MD5 was found insecure, so it’s recommended to use SHA-256 instead for better security.

References

- MD5 on Wikipedia
- Secure Hash Algorithm on Wikipedia
- Cryptographic hash function on Wikipedia
- Class MessageDigest Javadoc

Do you want to be expert in Java programming? If you do, why not join our mailing list to get advices from the professionals everyday? Just click here: http://newsletter.codejava.net - It's FREE, Quick and Awesome!

Attachments:

HashFunctionExample.zip [Java source code] 3 kB

Add comment

Name | E-mail

comment

1000 symbols left

Notify me of follow-up comments

Type the text

Privacy & Terms

Send

Comments

#6 cremildo come 2015-09-21 06:55 **0**

Hash Function Example, criptography.
[...]

Quote

#5 newkid 2015-08-04 05:25 **0**

well hello.. i m new to hashing and i have to write a java code on hashing for my project. the above code modules seem nice.but,can u tell me in what order to write them to obtain an executable program for sha256?... thanks in advance.

Quote

#4 angad 2015-02-05 00:29 **-1**

code is not working

Quote

#3 Ravindra Patil 2014-10-22 06:10 **0**

HashGenerationException is User Defined or System Defined please give me code of that one

Quote

#2 **Nam** 2014-09-06 22:17 **0**

Updated to use try-with-resources statement in the hashFile() method. Thank you for your suggestion. Good catch!

Quote

#1 **floj** 2014-07-15 00:36 **0**

Even if it is an example, in "2. Generating Hash from File" you should close the input stream in an finally block to prevent resource leaks. Or even better use the try-with-statement from java 7.

Quote

Refresh comments list

RSS feed for comments to this post

JComments

Copyright © 2012 - 2015 by www.codejava.net