

[Home](#)

[Java Core](#)

[Java SE](#)

[Java EE](#)

[Frameworks](#)

[IDEs](#)

[Servers](#)

[Coding](#)

[Books](#)

[Videos](#)

[Java Skills Test](#)



[Home](#) ▶ [Coding](#)



Free Java Tutorial Videos

Name:

Email:

Send Me

We respect your privacy

Featured Books

Java Coding Guideline Recommendations for Reliable and Secure Programs (SEI Series in Software Engineering)

Head First Design Patterns

Java Concurrency in Practice

Java Performance

Java Puzzlers: Traps, Pitfalls, and Corner Cases

Head First Object-Oriented Analysis and Design

Clean Code: A Handbook of Agile Software Craftsmanship

Algorithms Unlocked

Data Structures and Algorithm Analysis in Java (3rd Edition)

Refactoring: Improving the Design of Existing Code

The Pragmatic Programmer: From Journeyman to Master

Code Complete: A Practical Handbook of Software Construction, Second Edition

Cracking the Coding Interview: 150 Program Questions and Solutions

The Clean Coder: A Code of Conduct for Professional Programmers (Robert C. Martin Series)

Learn Java FREE with these 5 VIDEO COURSES

Your Name:

Your Email:

SEND ME THESE VIDEOS

We respect your privacy

How to calculate MD5 and SHA hash values in Java

Last Updated on 06 September 2014 | [Print](#) [Email](#)

Java Hosting from just \$5/mo. Get it [Download Aspose](#) for all your file format manipulation

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):

```
1 // algorithm can be "MD5", "SHA-1", "SHA-256"
2 MessageDigest digest = MessageDigest.getInstance(algorithm);
3
4 byte[] inputBytes = // get bytes array from message
5
6 byte[] hashBytes = digest.digest(inputBytes);
7
8 // convert hash bytes to string (usually in hexadecimal form)
```

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

```
1 MessageDigest digest = MessageDigest.getInstance(algorithm);
2
3 byte[] inputBytes = // get bytes array from message
4
5 digest.update(inputBytes);
6
7 byte[] hashedBytes = digest.digest();
8
9 // 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:

```
1 private static String hashString(String message, String algorithm)
2     throws HashGenerationException {
3
4     try {
5         MessageDigest digest = MessageDigest.getInstance(algorithm);
6         byte[] hashedBytes = digest.digest(message.getBytes("UTF-8"));
7
8         return convertByteArrayToHexString(hashedBytes);
9     } catch (NoSuchAlgorithmException | UnsupportedEncodingException ex) {
10         throw new HashGenerationException(
11             "Could not generate hash from String", ex);
12     }
13 }
```

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

```
1 private static String convertByteArrayToHexString(byte[] arrayBytes) {
2     StringBuffer stringBuffer = new StringBuffer();
3     for (int i = 0; i < arrayBytes.length; i++) {
4         stringBuffer.append(Integer.toString((arrayBytes[i] & 0xff) + 0x100
5             .substring(1)));
6     }
7     return stringBuffer.toString();
8 }
```

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

```
1 public static String generateMD5(String message) throws HashGenerationExce
2     return hashString(message, "MD5");
3 }
4
5 public static String generateSHA1(String message) throws HashGenerationExc
6     return hashString(message, "SHA-1");
7 }
8
9 public static String generateSHA256(String message) throws HashGenerationE
10    return hashString(message, "SHA-256");
11 }
```

Hence we have the following utility class:

```

1  package net.codejava.security;
2
3  import java.io.UnsupportedEncodingException;
4  import java.security.MessageDigest;
5  import java.security.NoSuchAlgorithmException;
6
7  /**
8   * Hash functions utility class.
9   * @author www.codejava.net
10  *
11  */
12  public class HashGeneratorUtils {
13      private HashGeneratorUtils() {
14
15      }
16
17      public static String generateMD5(String message) throws HashGeneration
18          return hashString(message, "MD5");
19  }
20
21      public static String generateSHA1(String message) throws HashGeneratic
22          return hashString(message, "SHA-1");
23  }
24
25      public static String generateSHA256(String message) throws HashGenerat
26          return hashString(message, "SHA-256");
27  }
28
29      private static String hashString(String message, String algorithm)
30          throws HashGenerationException {
31
32          try {
33              MessageDigest digest = MessageDigest.getInstance(algorithm);
34              byte[] hashedBytes = digest.digest(message.getBytes("UTF-8"));
35
36              return convertByteArrayToHexString(hashedBytes);
37          } catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {
38              throw new HashGenerationException(
39                  "Could not generate hash from String", e);
40          }
41      }
42
43      private static String convertByteArrayToHexString(byte[] arrayBytes) {
44          StringBuffer stringBuffer = new StringBuffer();
45          for (int i = 0; i < arrayBytes.length; i++) {
46              stringBuffer.append(Integer.toString((arrayBytes[i] & 0xff) +
47                  .substring(1));
48          }
49          return stringBuffer.toString();
50      }
51  }

```

Here's a test program:

```

1 package net.codejava.security;
2
3 /**
4  * Test generating hash values from String.
5  * @author www.codejava.net
6  *
7  */
8 public class StringHashGeneratorExample {
9
10     public static void main(String[] args) {
11         try {
12             String inputString = args[0];
13             System.out.println("Input String: " + inputString);
14
15             String md5Hash = HashGeneratorUtils.generateMD5(inputString);
16             System.out.println("MD5 Hash: " + md5Hash);
17
18             String sha1Hash = HashGeneratorUtils.generateSHA1(inputString);
19             System.out.println("SHA-1 Hash: " + sha1Hash);
20
21             String sha256Hash = HashGeneratorUtils.generateSHA256(inputString);
22             System.out.println("SHA-256 Hash: " + sha256Hash);
23         } catch (HashGenerationException ex) {
24             ex.printStackTrace();
25         }
26     }
27 }
28

```

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

```

1 Input String: admin
2 MD5 Hash: 21232f297a57a5a743894a0e4a801fc3
3 SHA-1 Hash: d033e22ae348aeb5660fc2140aec35850c4da997
4 SHA-256 Hash: 8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a

```

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:

```

1 private static String hashFile(File file, String algorithm)
2     throws HashGenerationException {
3     try (FileInputStream inputStream = new FileInputStream(file)) {
4         MessageDigest digest = MessageDigest.getInstance(algorithm);
5
6         byte[] bytesBuffer = new byte[1024];
7         int bytesRead = -1;
8
9         while ((bytesRead = inputStream.read(bytesBuffer)) != -1) {
10             digest.update(bytesBuffer, 0, bytesRead);
11         }
12
13         byte[] hashedBytes = digest.digest();
14
15         return convertByteArrayToHexString(hashedBytes);
16     } catch (NoSuchAlgorithmException | IOException ex) {
17         throw new HashGenerationException(
18             "Could not generate hash from file", ex);
19     }
20 }

```

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

```

1 public static String generateMD5(File file) throws HashGenerationException
2     return hashFile(file, "MD5");
3 }
4
5 public static String generateSHA1(File file) throws HashGenerationException
6     return hashFile(file, "SHA-1");
7 }
8
9 public static String generateSHA256(File file) throws HashGenerationException
10    return hashFile(file, "SHA-256");
11 }

```

And here's a test program:

```

1 package net.codejava.security;
2
3 import java.io.File;
4
5 /**
6  * Test generating hash values from File.
7  * @author www.codejava.net
8  *
9  */
10 public class FileHashGeneratorExample {
11
12     public static void main(String[] args) {
13         try {
14             String filePath = args[0];
15             System.out.println("File Path: " + filePath);
16             File file = new File(filePath);
17
18             String md5Hash = HashGeneratorUtils.generateMD5(file);
19             System.out.println("MD5 Hash: " + md5Hash);
20
21             String sha1Hash = HashGeneratorUtils.generateSHA1(file);
22             System.out.println("SHA-1 Hash: " + sha1Hash);
23
24             String sha256Hash = HashGeneratorUtils.generateSHA256(file);
25             System.out.println("SHA-256 Hash: " + sha256Hash);
26
27         } catch (HashGenerationException ex) {
28             ex.printStackTrace();
29         }
30     }
31 }
32

```

Example output:

```

1 File Path: D:\Java\PDFViewer\JPedalPDFViewer.zip
2 MD5 Hash: 56a86f56a18b73353e5f0afa7b142ed1
3 SHA-1 Hash: dc55bd7e84c4787242499ec068fa145bcca01937
4 SHA-256 Hash: 093059d79d009662a0a7f70c74cec934a73c1becc8ac813cdcc4995f2aeb8

```

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](#)

Share this article:



Free Java Tutorial Videos:

EMAIL ADDRESS:

you@domain.com

FIRST NAME:

John

SEND ME

Attachments:

 [HashFunctionExample.zip](#) [Java source code] 3 kB

Add comment

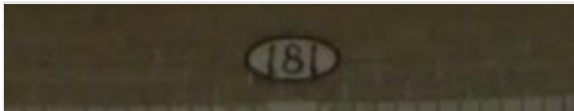
Name

E-mail

comment

500 symbols left

☐ Notify me of follow-up comments



reCAPTCHA™

Type the text

[Privacy & Terms](#)

Send

Comments

#6 **cremildo come** 2015-09-21 06:55
Hash Function Example, cryptography.
[...]

0

[Quote](#)

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

0

[Quote](#)

#4 **angad** 2015-02-05 00:29
code is not working

-1

[Quote](#)

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

+2

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

[JavaTipsEveryDay](#) [About](#) [Advertise](#) [Contribute](#) [Contact](#) [Terms of Use](#) [Privacy Policy](#) [Site Map](#) [Newsletter](#)



Copyright © 2012 - 2016 by www.codejava.net