Java – Generate Random String

Last modified: May 6, 2021

by Eugen Paraschiv (https://www.baeldung.com/author/eugen/)

Java (https://www.baeldung.com/category/java/) +

Java String (https://www.baeldung.com/tag/java-string/)

Random (https://www.baeldung.com/tag/random/)

1. Introduction

In this tutorial, we're going to learn how to generate a random string in Java, first using the standard Java libraries, then using a Java 8 variant, and finally using the Apache Commons Lang library (https://commons.apache.org/proper/commons-lang/).

This article is part of the "Java – Back to Basic" series (/java-tutorial) here on Baeldung.

2. Generate Random Unbounded String With Plain Java

Let's start simple and generate a random *String* bounded to 7 characters:

```
@Test
public void
givenUsingPlainJava_whenGeneratingRandomStringUnbounded_thenCorrect() {
    byte[] array = new byte[7]; // length is bounded by 7
    new Random().nextBytes(array);
    String generatedString = new String(array, Charset.forName("UTF-8"));

    System.out.println(generatedString);
}
```

Keep in mind that the new string will not be anything remotely alphanumeric.

Further reading:

Efficient Word Frequency Calculator in Java (/java-word-frequency)

Explore various ways of counting words in Java and see how they perform.

Read more (/java-word-frequency) →

Java – Random Long, Float, Integer and Double (/java-generate-random-long-float-integer-double)

Learn how to generate random numbers in Java - both unbounded as well as within a given interval.

Read more (/java-generate-random-long-float-integer-double) →

Guide to Java String Pool (/java-string-pool)

Learn how the JVM optimizes the amount of memory allocated to String storage in the Java String Pool.

Read more (/java-string-pool) →

3. Generate Random Bounded String With Plain Java

Next let's look at creating a more constrained random string; we're going to generate a random *String* using lowercase alphabetic letters and a set length:

```
@Test
public void
givenUsingPlainJava_whenGeneratingRandomStringBounded_thenCorrect() {

    int leftLimit = 97; // letter 'a'
    int rightLimit = 122; // letter 'z'
    int targetStringLength = 10;
    Random random = new Random();
    StringBuilder buffer = new StringBuilder(targetStringLength);
    for (int i = 0; i < targetStringLength; i++) {
        int randomLimitedInt = leftLimit + (int)
          (random.nextFloat() * (rightLimit - leftLimit + 1));
        buffer.append((char) randomLimitedInt);
    }
    String generatedString = buffer.toString();

    System.out.println(generatedString);
}
```

4. Generate Random Alphabetic String With Java 8

Now let's use *Random.ints*, added in JDK 8, to generate an alphabetic *String*:

```
@Test
public void
givenUsingJava8_whenGeneratingRandomAlphabeticString_thenCorrect() {
    int leftLimit = 97; // letter 'a'
    int rightLimit = 122; // letter 'z'
    int targetStringLength = 10;
    Random random = new Random();

    String generatedString = random.ints(leftLimit, rightLimit + 1)
      .limit(targetStringLength)
      .collect(StringBuilder::new, StringBuilder::appendCodePoint,
StringBuilder::append)
      .toString();

    System.out.println(generatedString);
}
```

5. Generate Random Alphanumeric String With Java 8

Then we can widen our character set in order to get an alphanumeric *String:*

```
@Test
public void
givenUsingJava8_whenGeneratingRandomAlphanumericString_thenCorrect() {
    int leftLimit = 48; // numeral '0'
    int rightLimit = 122; // letter 'z'
    int targetStringLength = 10;
    Random random = new Random();

    String generatedString = random.ints(leftLimit, rightLimit + 1)
      .filter(i -> (i <= 57 || i >= 65) && (i <= 90 || i >= 97))
      .limit(targetStringLength)
      .collect(StringBuilder::new, StringBuilder::appendCodePoint,
StringBuilder::append)
      .toString();

    System.out.println(generatedString);
}
```

We used the *filter* method above to leave out Unicode characters between 65 and 90 in order to avoid out of range characters.

6. Generate Bounded Random String With Apache Commons Lang

The Commons Lang library from Apache helps a lot with random string generation. Let's take a look at **generating a bounded *String* using only letters**:

(h
tt

```
@Test
public void
givenUsingApache_whenGeneratingRandomStringBounded_thenCorrect() {

    int length = 10;
    boolean useLetters = true;
    boolean useNumbers = false;
    String generatedString = RandomStringUtils.random(length, useLetters,
useNumbers);

    System.out.println(generatedString);
}
```

So instead of all the low-level code in the Java example, this one is done with a simple one-liner.

7. Generate Alphabetic String With Apache Commons Lang

Here is another very simple example, this time a bounded *String* with only alphabetic characters, but without passing boolean flags into the API:

```
@Test
public void
givenUsingApache_whenGeneratingRandomAlphabeticString_thenCorrect() {
    String generatedString = RandomStringUtils.randomAlphabetic(10);

    System.out.println(generatedString);
}
```

8. Generate Alphanumeric String With Apache Commons Lang

Finally, we have the same random bounded *String,* but this time numeric:

```
@Test
public void
givenUsingApache_whenGeneratingRandomAlphanumericString_thenCorrect() {
    String generatedString = RandomStringUtils.randomAlphanumeric(10);

    System.out.println(generatedString);
}
```

And there we have it, **creating bounded and unbounded strings** with either plain Java, a Java 8 variant, or the Apache Commons Library.

9. Conclusion

Through different implementation methods, we were able to generate bound and unbound strings using plain Java, a Java 8 variant, or the Apache Commons Library.

(h
tt

In these Java examples, we used *java.util.Random*, but one point worth mentioning is that it is not cryptographically secure. **Consider using *java.security.SecureRandom* (/java-secure-random) instead for security-sensitive applications.**

The implementation of all of these examples and snippets can be found in the GitHub project (https://github.com/eugenp/tutorials/tree/master/core-java-modules/core-java-strings). This is a Maven-based project so it should be easy to import and run.

Learning to "Build your API **with Spring**"?

Enter your email address

>> Get the eBook

4 COMMENTS

Oldest

View Comments

Comments are closed on this article!

CATEGORIES

SPRING (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
REST (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SECURITY (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
HTTP CLIENT-SIDE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

SERIES

JAVA "BACK TO BASICS" TUTORIAL (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
REST WITH SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
SPRING PERSISTENCE TUTORIAL (/PERSISTENCE-WITH-SPRING-SERIES)
SECURITY WITH SPRING (/SECURITY-SPRING)
SPRING REACTIVE TUTORIALS (/SPRING-REACTIVE-GUIDE)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE COURSES (HTTPS://COURSES.BAELDUNG.COM)
JOBS (/TAG/ACTIVE-JOB/)
THE FULL ARCHIVE (/FULL_ARCHIVE)
WRITE FOR BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORS (/EDITORS)
OUR PARTNERS (/PARTNERS)
ADVERTISE ON BAELDUNG (/ADVERTISE)

TERMS OF SERVICE (/TERMS-OF-SERVICE)

PRIVACY POLICY (/PRIVACY-POLICY)

COMPANY INFO (/BAELDUNG-COMPANY-INFO)

CONTACT (/CONTACT)