Apache Commons CLI

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

The Apache Commons CLI are the components of the Apache Commons which are derived from Java API and provides an API to parse command line arguments/options which are passed to the programs. This API also enables to print the help related to options available. This tutorial covers most of the topics required for a basic understanding of Apache Commons CLI and to get a feel of how it works.

Audience

This tutorial has been prepared for the beginners to help them understand the basic to advanced concepts related to the Apache Commons CLI.

Prerequisites

Before you start practicing various types of examples given in this reference, we assume that you are already aware about computer programs and computer programming languages.

Copyright & Disclaimer

Table of Contents

1. Apache Commons CLI — Overview

The Apache Commons CLI are the components of the Apache Commons which are derived from Java API and provides an API to parse command line arguments/options which are passed to the programs. This API also enables to print help related to options available.

Command line processing comprises of three stages. These stages are explained below:

- Definition Stage
- Parsing Stage
- Interrogation Stage

Definition Stage

In definition stage, we define the options that an application can take and act accordingly. Commons CLI provides Options class, which is a container for Option objects.

```
// create Options object
Options options = new Options();


// add a option
options.addOption("a", false, "add two numbers");
```

Here we have added an option flag **a**, while false as second parameter, signifies that option is not mandatory and third parameter states the description of option.

Parsing Stage

In parsing stage, we parse the options passed using command line arguments after creating a parser instance.

```
//Create a parser
CommandLineParser parser = new DefaultParser();


//parse the options passed as command line arguments
CommandLine cmd = parser.parse( options, args);
```

Interrogation Stage

In Interrogation stage, we check if a particular option is present or not and then, process the command accordingly.

```
//hasOptions checks if option is present or not
```

```
if(cmd.hasOption("a")) {
   // add the two numbers
} else if(cmd.hasOption("m")) {
   // multiply the two numbers
}
```

2. Apache Commons CLI — Environment Setup

In this chapter, we will learn about the local environment setup of Apache Commons CLI and how to set up the path of Commons CLI for Windows 2000/XP, Windows 95/98/ME etc. We will also understand about some popular java editors and how to download Commons CLI archive.

Local Environment Setup

If you are still willing to set up your environment for Java programming language, then this chapter will guide you on how to download and set up Java on your machine. Please follow the steps mentioned below to set up the environment.

Java SE is freely available from the link https://www.oracle.com/java/technologies/oracle-java-archive-downloads.html. So you can download a version based on your operating system.

Follow the instructions to download Java and run the **.exe** to install Java on your machine. Once you have installed Java on your machine, you would need to set environment variables to point to correct installation directories.

Path for Windows 2000/XP

We are assuming that you have installed Java in **c:\Program Files\java\jdk** directory.

- Right-click on **'My Computer'** and select **'Properties'**.
- Click on the **'Environment variables'** button under the **'Advanced'** tab.
- Now, alter the **'Path'** variable, so that it also contains the path to the Java executable. Example, if the path is currently set to **'C:\WINDOWS\SYSTEM32'**, then change your path to read **'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'**.

Path for Windows 95/98/ME

We are assuming that you have installed Java in **c:\Program Files\java\jdk** directory.

- Edit the **'C:\autoexec.bat'** file and add the following line at the end – **'SET PATH=%PATH%;C:\Program Files\java\jdk\bin'.**

Path for Linux, UNIX, Solaris, FreeBSD

Environment variable PATH should be set to point, where the Java binaries have been installed. Refer to your shell documentation, if you have trouble doing this.

Example, if you use bash as your shell, then you would add the following line to the end of your '.bashrc: export PATH=/path/to/java:$PATH'

Popular Java Editors

To write your Java programs, you need a text editor. There are many sophisticated IDEs available in the market. But for now, you can consider one of the following:

- **Notepad** – On Windows machine you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans** – It is a Java IDE that is open-source and free which can be downloaded from https://www.netbeans.org/index.html.
- **Eclipse** – It is also a Java IDE developed by the eclipse open-source community and can be downloaded from https://www.eclipse.org/.

Download Common CLI Archive

Download the latest version of Apache Common CLI jar file from commons-cli-1.4-bin.zip. At the time of writing this tutorial, we have downloaded *commons-cli-1.4-bin.zip* and copied it into C:\>Apache folder.

OS	Archive name
Windows	commons-cli-1.4-bin.zip
Linux	commons-cli-1.4-bin.tar.gz
Mac	commons-cli-1.4-bin.tar.gz

Apache Common CLI Environment

Set the **APACHE_HOME** environment variable to point to the base directory location where, Apache jar is stored on your machine. Assume that we have extracted commons-collections4-4.1-bin.zip in Apache folder on various Operating Systems as follows:

OS	Output
Windows	Set the environment variable APACHE_HOME to C:\Apache
Linux	export APACHE_HOME=/usr/local/Apache
Mac	export APACHE_HOME=/Library/Apache

CLASSPATH Variable

Set the **CLASSPATH** environment variable to point to the Common CLI jar location. Assume that you have stored commons-cli-1.4.jar in Apache folder on various Operating Systems as follows:

OS	Output
Windows	Set the environment variable CLASSPATH to %CLASSPATH%;%APACHE_HOME%\commons-cli-1.4.jar;.;
Linux	export CLASSPATH=$CLASSPATH:$APACHE_HOME/commons-cli-1.4.jar:.
Mac	export CLASSPATH=$CLASSPATH:$APACHE_HOME/commons-cli-1.4.jar:.

3. Apache Commons CLI — First Application

Let's create a sample console based application, whose purpose is to get either sum of passed numbers or multiplication of passed numbers based on the options used.

Create a java class named CLITester.

Example

CLITester.java

```
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.DefaultParser;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class CLITester {
   public static void main(String[] args) throws ParseException {

      //***Definition Stage***
      // create Options object
      Options options = new Options();

      // add option "-a"
      options.addOption("a", false, "add numbers");

      // add option "-m"
      options.addOption("m", false, "multiply numbers");

      //***Parsing Stage***
      //Create a parser
      CommandLineParser parser = new DefaultParser();

      //parse the options passed as command line arguments
       CommandLine cmd = parser.parse( options, args);

       //***Interrogation Stage***
      //hasOptions checks if option is present or not
      if(cmd.hasOption("a")) {
         System.out.println("Sum of the numbers: " + getSum(args));
      } else if(cmd.hasOption("m")) {
         System.out.println("Multiplication of the numbers: " + 
getMultiplication(args));
      }
   }

   public static int getSum(String[] args) {
      int sum = 0;
      for(int i = 1; i < args.length ; i++) {
         sum += Integer.parseInt(args[i]);
      }
      return sum;
   }

   public static int getMultiplication(String[] args) {
      int multiplication = 1;
      for(int i = 1; i < args.length ; i++) {
         multiplication *= Integer.parseInt(args[i]);
      }
      return multiplication;
   }
}
```

Output

Run the file, while passing -a as option and numbers to get the sum of the numbers as result.

```
java CLITester -a 1 2 3 4 5
Sum of the numbers: 15
```

Run the file, while passing -m as option and numbers to get the multiplication of the numbers as result.

```
java CLITester -m 1 2 3 4 5
Multiplication of the numbers: 120
```

4. Apache Commons CLI — Option Properties

Option object is used to represent the Option passed to command line program. Following are various properties that an Option object possess.

Sr.No.	Name (Type) & Description
1	**opt (String)** Identification string of the Option.
2	**longOpt (String)** Alias and more descriptive identification string.
3	**description (String)** Description of the function of the option.
4	**required (boolean)** Flag to check whether the option must appear on the command line.
5	**arg (boolean)** Flag to check whether the option takes an argument.
6	**args (boolean)** Flag to check whether the option takes more than one argument.
7	**optionalArg (boolean)** Flag to check whether the option's argument is optional.
8	**argName (String)** Name of the argument value for the usage statement.
9	**valueSeparator (char)** The character value used to split the argument string.
10	**type (Object)** Argument type.

11	**value (String)** Option value.
12	**values (String[])** Values of the option.

5. Apache Commons CLI — Boolean Option

A boolean option is represented on a command line by its presence. For example, if option is present, then its value is true, otherwise, it is considered as false. Consider the following example, where we are printing current date and if -t flag is present. Then, we will print time too.

Example

CLITester.java

```
import java.util.Calendar;
import java.util.Date;

import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.DefaultParser;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class CLITester {
   public static void main(String[] args) throws ParseException {
      Options options = new Options();
      options.addOption("t", false, "display time");

      CommandLineParser parser = new DefaultParser();
      CommandLine cmd = parser.parse( options, args);

      Calendar date = Calendar.getInstance();
      int day = date.get(Calendar.DAY_OF_MONTH);
      int month = date.get(Calendar.MONTH);
      int year = date.get(Calendar.YEAR);

      int hour = date.get(Calendar.HOUR);
      int min = date.get(Calendar.MINUTE);
      int sec = date.get(Calendar.SECOND);

      System.out.print(day + "/" + month + "/" + year);
      if(cmd.hasOption("t")) {
         System.out.print(" " + hour + ":" + min + ":" + sec);
      }
   }
}
```

Output

Run the file without passing any option and see the result.

```
java CLITester
12/11/2017
```

Run the file, while passing -t as option and see the result.

```
java CLITester
12/11/2017 4:13:10
```

6. Apache Commons CLI — Argument Option

An Argument option is represented on a command line by its name and its corresponding value. For example, if option is present, then user has to pass its value. Consider the following example, if we are printing logs to some file, for which, we want user to enter name of the log file with the argument option logFile.

Example

CLITester.java

```
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.DefaultParser;
import org.apache.commons.cli.Option;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class CLITester {
   public static void main(String[] args) throws ParseException {

      Options options = new Options();

      Option logfile   = Option.builder()
         .longOpt("logFile")
         .argName("file" )
         .hasArg()
         .desc("use given file for log" )
         .build();

      options.addOption(logfile);

      CommandLineParser parser = new DefaultParser();
      CommandLine cmd = parser.parse( options, args);

      // has the logFile argument been passed?
      if(cmd.hasOption("logFile")) {
         //get the logFile argument passed
         System.out.println( cmd.getOptionValue( "logFile" ) );
      }
   }
}
```

Output

Run the file, while passing --logFile as option, name of the file as value of the option and see the result.

```
java CLITester --logFile test.log
test.log
```

7. Apache Commons CLI — Properties Option

A Properties option is represented on a command line by its name and its corresponding properties like syntax, which is similar to java properties file. Consider the following example, if we are passing options like -DrollNo=1 -Dclass=VI -Dname=Mahesh, we should process each value as properties. Let's see the implementation logic in action.

Example

CLITester.java

```
import java.util.Properties;


import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.DefaultParser;
import org.apache.commons.cli.Option;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;


public class CLITester {
   public static void main(String[] args) throws ParseException {


      Options options = new Options();


      Option propertyOption   = Option.builder()
         .longOpt("D")
         .argName("property=value" )
         .hasArgs()
         .valueSeparator()
         .numberOfArgs(2)
         .desc("use value for given properties" )
         .build();


      options.addOption(propertyOption);


      CommandLineParser parser = new DefaultParser();
      CommandLine cmd = parser.parse( options, args);
      if(cmd.hasOption("D")) {
         Properties properties = cmd.getOptionProperties("D");
         System.out.println("Class: " + properties.getProperty("class"));
         System.out.println("Roll No: " + properties.getProperty("rollNo"));
         System.out.println("Name: " + properties.getProperty("name"));
      }
   }
}
```

Output

Run the file, while passing options as key value pairs and see the result.

```
java CLITester -DrollNo=1 -Dclass=VI -Dname=Mahesh
Class: VI
Roll No: 1
Name: Mahesh
```

8. Apache Commons CLI —Posix Parser

A Posix parser is use to parse Posix like arguments passed. It is now deprecated and is replaced by DefaultParser.

Example

CLITester.java

```
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;
import org.apache.commons.cli.PosixParser;

public class CLITester {
   public static void main(String[] args) throws ParseException {

      //Create posix like options
      Options posixOptions = new Options();
      posixOptions.addOption("D", false, "Display");
      posixOptions.addOption("A", false, "Act");

      CommandLineParser posixParser = new PosixParser();

      CommandLine cmd = posixParser.parse(posixOptions, args);

      if( cmd.hasOption("D") ) {
         System.out.println("D option was used.");
      }

      if( cmd.hasOption("A") ) {
         System.out.println("A option was used.");
      }
   }
}
```

Output

Run the file while passing -D -A as options and see the result.

```
java CLITester -D -A

D option was used.

A option was used.
```

Run the file while passing --D as option and see the result.

```
java CLITester --D

D option was used.
```

9. Apache Commons CLI — GNU Parser

A GNU parser is use to parse gnu like arguments passed. It is now deprecated and is replaced by DefaultParser.

Example

CLITester.java

```java
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.CommandLineParser;
import org.apache.commons.cli.GnuParser;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class CLITester {
   public static void main(String[] args) throws ParseException {

      //Create GNU like options
      Options gnuOptions = new Options();
      gnuOptions.addOption("p", "print", false, "Print")
         .addOption("g", "gui", false, "GUI")
         .addOption("n", true, "Scale");

      CommandLineParser gnuParser = new GnuParser();
      CommandLine cmd = gnuParser.parse(gnuOptions, args);

      if( cmd.hasOption("p") ) {
         System.out.println("p option was used.");
      }

      if( cmd.hasOption("g") ) {
         System.out.println("g option was used.");
      }

      if( cmd.hasOption("n") ) {
         System.out.println("Value passed: " + cmd.getOptionValue("n"));
      }
   }
}
```

Output

Run the file while passing -p -g -n 10 as option and see the result.

```
java CLITester -p -g -n 10
p option was used.
g option was used.
Value passed: 10
```

10. Apache Commons CLI — Usage Example

Apache Commons CLI provides HelpFormatter class to print the usage guide of command line arguments. See the example given below:

Example

CLITester.java

```
import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class CLITester {
   public static void main(String[] args) throws ParseException {

      Options options = new Options();
      options.addOption("p", "print", false, "Send print request to printer.")
         .addOption("g", "gui", false, "Show GUI Application")
         .addOption("n", true, "No. of copies to print");

      HelpFormatter formatter = new HelpFormatter();
      formatter.printHelp("CLITester", options);
   }
}
```

Output

Run the file and see the result.

```
java CLITester
usage: CLITester
 -g,--gui      Show GUI Application
 -n <arg>      No. of copies to print
 -p,--print    Send print request to printer.
```

11. Apache Commons CLI — Help Example

Apache Commons CLI provides HelpFormatter class to print the help related to command line arguments. See the example.

Example

CLITester.java

```
import java.io.PrintWriter;

import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Options;
import org.apache.commons.cli.ParseException;

public class CLITester {
   public static void main(String[] args) throws ParseException {

      Options options = new Options();
      options.addOption("p", "print", false, "Send print request to printer.")
         .addOption("g", "gui", false, "Show GUI Application")
         .addOption("n", true, "No. of copies to print");

      HelpFormatter formatter = new HelpFormatter();

      final PrintWriter writer = new PrintWriter(System.out);
      formatter.printUsage(writer,80,"CLITester", options);
      writer.flush();
   }
}
```

Output

Run the file and see the result.

```
java CLITester
usage: CLITester [-g] [-n <arg>] [-p]
```