Java.lang.ProcessBuilder class in Java

This class is used to create operating system processes.

Each ProcessBuilder instance manages a collection of process attributes. **The start() method creates a new Process instance with those attributes.** The start() method can be invoked repeatedly **from the same instance** to **create new subprocesses with identical or related attributes.**

ProcessBuilder can be used to help create operating system process.

Before JDK 5.0, the only way to create a process and execute it was to use Runtime.exec() method.

It extends class Object.

This class is not synchronised.

Constructor:

ProcessBuilder(List command): This constructs a process builder with the specified operating system program and arguments.

ProcessBuilder(String… command): This constructs a process builder with the specified operating system program and arguments.

Methods:

1/ List command(): This method returns the process builder's operating system program and arguments.

Syntax: public List command().

Returns: **this process builder's program and its arguments.**

Exceptions: **NullPointerException** - if the argument is null.

// Java code illustrating command() method

```
import java.lang.*;

import java.io.*;

class ProcessBuilderDemo
```

```
{
    public static void main(String[] arg) throws IOException
    {
        // creating list of process
        List<String> list = new ArrayList<String>();
        list.add("notepad.exe");

        // create the process
        ProcessBuilder build = new ProcessBuilder(list);

        // checking the command i list
        System.out.println("command: " + build.command());

    }
}
```

Output:

```
command: [notepad.exe]
```

2/ ProcessBuilder command(List command): This method sets the process builder’s operating system program and arguments.

Syntax: public ProcessBuilder command(List command).

Returns: NA.

Exception: NullPointerException - if the argument is null.

// Java code illustrating ProcessBuilder command(List<String> command)

```
import java.lang.*;
import java.io.*;
import java.util.*;
class ProcessBuilderDemo
{
    public static void main(String[] arg) throws IOException
```

```
    {
        // creating list of process
        List<String> list = new ArrayList<String>();
        list.add("notepad.exe");
        list.add("xyz.txt");

        // create the process
        ProcessBuilder build = new ProcessBuilder(list);

        // checking the command in list
        System.out.println("command: " + build.command());

    }
}
```

Output:

```
command: [notepad.exe, xyz.txt]
```

3/ ProcessBuilder directory(File directory): This method sets the process builder's working directory. Subprocesses subsequently started by object's **start() method will use it as their working directory.** The argument may be null – which means to use the working directory of the current Java process, usually the directory named by the system property user.dir, as the working directory of the child process.

Syntax: public ProcessBuilder directory(File directory).

Returns: this process builder.

Exception: NA.

// Java code illustrating directory() method

```
import java.lang.*;
import java.io.*;
import java.util.*;
class ProcessBuilderDemo
{
```

```
    public static void main(String[] arg) throws IOException
    {
        // creating list of process
        List<String> list = new ArrayList<String>();
        list.add("notepad.exe");
        list.add("abc.txt");

        // creating the process
        ProcessBuilder build = new ProcessBuilder(list);

        // setting the directory
        build.directory(new File("src"));

        // checking the directory, on which currently working on
        System.out.println("directory: " + build.directory());

    }
}
```

Output:

directory: src

4/ Map environment(): This method returns a string map view of the process builder's environment. Whenever a process builder is created, the environment is initialized to a copy of the current process environment. Subprocesses subsequently **started by the object's start() method will use this map as their environment.**

Syntax: public Map environment()

Returns: this process builder's environment

Exception: **SecurityException** - if a security manager exists and its checkPermission method doesn't allow access to the process environment.

// Java code illustrating environment() method

```
import java.io.*;
```

```
import java.util.*;
class ProcessBuilderDemo
{
    public static void main(String[] arg) throws IOException
    {
        // creating the process
        ProcessBuilder pb = new ProcessBuilder();

        // map view of this process builder's environment
        Map<String, String> envMap = pb.environment();

        // checking map view of environment
        for(Map.Entry<String, String> entry : envMap.entrySet())
        {.
             // checking key and value separately
            System.out.println("Key = " + entry.getKey() +
               ", Value = " + entry.getValue());
        }
    }
}
```

Output:

```
Key = PATH, Value = /usr/bin:/bin:/usr/sbin:/sbin
Key = JAVA_MAIN_CLASS_14267, Value = ProcessBuilderDemo
Key = J2D_PIXMAPS, Value = shared
Key = SHELL, Value = /bin/bash
Key = JAVA_MAIN_CLASS_11858, Value = org.netbeans.Main
Key = USER, Value = abhishekverma
Key = TMPDIR, Value =
/var/folders/9z/p63ysmfd797clc0468vvy4980000gn/T/
Key = SSH_AUTH_SOCK, Value =
/private/tmp/com.apple.launchd.uWvCfYQWBP/Listeners
Key = XPC_FLAGS, Value = 0x0
```

Key = LD_LIBRARY_PATH, Value = /Library/Java/JavaVirtualMachines

/jdk1.8.0_121.jdk/Contents/Home/jre/lib/

amd64:/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk

/Contents/Home/jre/lib/i386:

Key = __CF_USER_TEXT_ENCODING, Value = 0x1F5:0x0:0x0

Key = Apple_PubSub_Socket_Render, Value = /private/tmp/com.apple.launchd.weuNq4pAfF/Render

Key = LOGNAME, Value = abhishekverma

Key = LC_CTYPE, Value = UTF-8

Key = XPC_SERVICE_NAME, Value = 0

Key = PWD, Value = /

Key = SHLVL, Value = 1

Key = HOME, Value = /Users/abhishekverma

Key = _, Value = /Library/Java/JavaVirtualMachines/

jdk1.8.0_121.jdk/Contents/Home/bin/java

Understanding the above output: The output **is map view of the subprocess build upon**. This above output vary from user to user totally depends upon the operating system and user.

5/ boolean redirectErrorStream(): **This method tells whether the process builder merges standard error and standard output**. If this property is true, then any error output generated by subprocesses subsequently started by the object's start() method **will be merged with the standard output, so that both can be read using the Process.getInputStream()** method. It makes it easier to correlate error messages with the corresponding output. **The initial value is false.**

Syntax: public boolean redirectErrorStream().

Returns: this process builder's redirectErrorStream property.

Exception: NA.

// Java code illustrating redirectErrorStream() method

```
import java.lang.*;

import java.io.*;

import java.util.*;

class ProcessBuilderDemo
```

```
{
    public static void main(String[] arg) throws IOException
    {
         // creating list of commands
        List list = new ArrayList();

        list.add("notepad.exe");
        list.add("xyz.txt");

        //creating the process
        ProcessBuilder build = new ProcessBuilder(list);

        // checking if error stream is redirected
        System.out.println(build.redirectErrorStream());

    }
}
```

Output:

false

6/ ProcessBuilder redirectErrorStream(boolean redirectErrorStream): This method **sets this process builder's redirectErrorStream property.** If this property is true, then any error output generated by subprocesses subsequently started by this object's start() method **will be merged with the standard output, so that both can be read using the Process.getInputStream() method.** This makes it easier to correlate error messages with the corresponding output. The initial value is **false.**

Syntax: public ProcessBuilder redirectErrorStream(boolean redirectErrorStream).

Returns: this process builder.

Exception: NA.

// Java code illustrating redirectErrorStream(boolean redirectErrorStream)

```
// method
import java.lang.*;
import java.io.*;
import java.util.*;
class ProcessBuilderDemo
{
    public static void main(String[] arg) throws IOException
    {
        // creating list of commands
        List list = new ArrayList();

        list.add("notepad.exe");
        list.add("xyz.txt");

        //creating the process
        ProcessBuilder build = new ProcessBuilder(list);

        // redirecting error stream
        build.redirectErrorStream(true);

        // checking if error stream is redirected
        System.out.println(build.redirectErrorStream());

     }
}
```

Output:

true

7/ Process start(): This method starts a new process using the attributes of process builder. The new process will invoke the command and arguments given by **command**(), in a working directory as given by **directory**(), with a process environment as given by **environment**(). This method checks that

the command is a valid operating system command. Which commands are valid is system-dependent, but at the very least the command must be a non-empty list of non-null strings.

If there is a security manager, its checkExec method is called with the first component of this object's command array as its argument. It may result in a **SecurityException being thrown.**

Syntax: public Process start().

Returns: **a new Process object for managing the subprocess.**

Exception: **NullPointerException** - If an element of the command list is null

IndexOutOfBoundsException - If the command is an empty list (has size 0).

SecurityException - If a security manager exists and its checkExec method doesn't allow creation of the subprocess.

IOException - If an I/O error occurs.

// Java code illustrating start() method

```
import java.lang.*;

import java.io.*;

import java.util.*;

class ProcessBuilderDemo

{

    public static void main(String[] arg) throws IOException

    {

        // creating list of commands

        List<String> commands = new ArrayList<String>();

        commands.add("ls"); // command

        commands.add("-l"); // command

        commands.add("/Users/abhishekverma"); //command in Mac OS


        // creating the process

        ProcessBuilder pb = new ProcessBuilder(commands);


        // startinf the process
```

```
        Process process = pb.start();

        // for reading the ouput from stream
        BufferedReader stdInput = new BufferedReader(new
         InputStreamReader(process.getInputStream()));
        String s = null;
        while ((s = stdInput.readLine()) != null)
        {
            System.out.println(s);
        }
    }
}
```

Output:

```
total 0
drwxr-xr-x  10 abhishekverma  staff   340 Jun 20 02:24 AndroidStudioProjects
drwx------@ 22 abhishekverma  staff   748 Jun 20 03:00 Desktop
drwx------@  7 abhishekverma  staff   238 Apr 29 22:03 Documents
drwx------+ 27 abhishekverma  staff   918 Jun 20 03:01 Downloads
drwx------@ 65 abhishekverma  staff  2210 Jun 18 20:48 Library
drwx------+  3 abhishekverma  staff   102 Mar 28 13:08 Movies
drwx------+  4 abhishekverma  staff   136 Apr  8 04:51 Music
drwxr-xr-x   4 abhishekverma  staff   136 Jun 19 15:01 NetBeansProjects
drwx------+  5 abhishekverma  staff   170 Apr 10 09:46 Pictures
drwxr-xr-x+  6 abhishekverma  staff   204 Jun 18 20:45 Public
-rw-r--r--   1 abhishekverma  staff     0 Apr 15 19:23 newreactjs.jsx
```

Important point: The command used in above program is Mac OS command, system command vary from one operating system to other operating system.

8/ ProcessBuilder inheritIO(): Sets the **source and destination for subprocess standard I/O to be the same as those of the current Java process.**

Syntax: public ProcessBuilder inheritIO().

Returns: this process builder.

Exception: NA.

// Java code illustrating inheritIO() method

```
import java.lang.*;
import java.io.*;
import java.util.*;
class ProcessBuilderDemo
{
    public static void main(String[] arg) throws IOException,
            InterruptedException
    {
        ProcessBuilder pb = new ProcessBuilder
                ("echo", "Hello GeeksforGeeks\n"
                + "This is ProcessBuilder Example");
        pb.inheritIO();
        Process process = pb.start();
        process.waitFor();

    }
}
```

Output:

```
Hello GeeksforGeeks
This is ProcessBuilder Example
```

This article is contributed by Abhishek Verma. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.