

JAVA.LANG.THREADGROUP.ENUMERATE METHOD

http://www.tutorialspoint.com/java/lang/threadgroup_enumerate_threadgroup.htm

Copyright © tutorialspoint.com

Description

The **java.lang.ThreadGroup.enumerate***ThreadGroup[]list* method copies into the specified array references to every active subgroup in this thread group.

Declaration

Following is the declaration for **java.lang.ThreadGroup.enumerate** method

```
public int enumerate(ThreadGroup[] list)
```

Parameters

- **list** -- This is an array into which to place the list of thread groups.

Return Value

This method returns the number of thread groups put into the array.

Exception

- **SecurityException** -- if the current thread does not have permission to enumerate this thread group.

Example

The following example shows the usage of java.lang.ThreadGroup.enumerate method.

```
package com.tutorialspoint;

import java.lang.*;

public class ThreadGroupDemo implements Runnable
{
    public static void main(String[] args) {
        ThreadGroupDemo tg = new ThreadGroupDemo();
        tg.func();
    }

    public void func() {
        try {
            // create a parent ThreadGroup
            ThreadGroup pGroup = new ThreadGroup("Parent ThreadGroup");

            // create a child ThreadGroup for parent ThreadGroup
            ThreadGroup cGroup = new ThreadGroup(pGroup, "Child ThreadGroup");

            // create a thread
            Thread t1 = new Thread(pGroup, this);
            System.out.println("Starting " + t1.getName() + "...");
            t1.start();

            // create another thread
            Thread t2 = new Thread(cGroup, this);
            System.out.println("Starting " + t2.getName() + "...");
            t2.start();

            // returns the number of thread groups put into the array
            ThreadGroup[] grpList = new ThreadGroup[pGroup.activeGroupCount()];
            int count = pGroup.enumerate(grpList);
            for (int i = 0; i < count; i++) {
```

```

        System.out.println("ThreadGroup " + grpList[i].getName() +
            " found");
    }

    // block until the other threads finish
    t1.join();
    t2.join();
}
catch (InterruptedException ex) {
    System.out.println(ex.toString());
}
}

// implements run()
public void run() {

    for(int i = 0; i < 1000; i++) {
        i++;
    }
    System.out.println(Thread.currentThread().getName() +
        " finished executing.");
}
}

```

Let us compile and run the above program, this will produce the following result:

```

Starting Thread-0...
Starting Thread-1...
ThreadGroup Child ThreadGroup found
Thread-0 finished executing.
Thread-1 finished executing.

```

Loading [Mathjax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js