**Java Thread Pool**

**Java Thread pool** represents a group of worker threads that are waiting for the job and reuse many times.

In case of thread pool, a group of fixed size threads are created. A thread from the thread pool is pulled out and assigned a job by the service provider. After completion of the job, thread is contained in the thread pool again.

**Advantage of Java Thread Pool**

**Better performance** It saves time because there is no need to create new thread.

**Real time usage**

It is used in Servlet and JSP where container creates a thread pool to process the request.

**Example of Java Thread Pool**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.io.\*; **import** java.util.Scanner; **import** java.util.concurrent.ExecutorService; **import** java.util.concurrent.Executors;  **public class** Main {   **public static void** main(String[] args){  ExecutorService executorService = Executors.*newFixedThreadPool*(5); *//creating a pool of 5 threads* **for** (**int** i = 0; i <= 10; i++){  Runnable worker = **new** Multis(**" "** + i);  executorService.execute(worker); *//calling execute method of ExecutorService* }   executorService.shutdown();  **while** (!executorService.isTerminated()){   }   System.***out***.println(**"all threads are finished"**);  } } | **package** com.company;  **import** java.util.Date;  **public class** Multis **extends** Thread{   **private** String **message**;   **public** Multis (String s){  **message** = s;  }   @Override  **public void** run(){  System.***out***.println(*currentThread*().getName() + **" (Start message) "** + **message**);  processHandler(); *//call processmessage method that sleeps the thread for 2 seconds* System.***out***.println(*currentThread*().getName() + **" (End)"**); *//prints thread name* }   **private void** processHandler(){  **try** {  Thread.*sleep*(2000);  } **catch** (Exception e){  System.***out***.println(**"Message : "** + e.getStackTrace());  }  } } |

pool-1-thread-5 (Start message) 4

pool-1-thread-4 (Start message) 3

pool-1-thread-3 (Start message) 2

pool-1-thread-2 (Start message) 1

pool-1-thread-1 (Start message) 0

pool-1-thread-5 (End)

pool-1-thread-5 (Start message) 5

pool-1-thread-4 (End)

pool-1-thread-4 (Start message) 6

pool-1-thread-3 (End)

pool-1-thread-3 (Start message) 7

pool-1-thread-2 (End)

pool-1-thread-2 (Start message) 8

pool-1-thread-1 (End)

pool-1-thread-1 (Start message) 9

pool-1-thread-5 (End)

pool-1-thread-5 (Start message) 10

pool-1-thread-4 (End)

pool-1-thread-3 (End)

pool-1-thread-2 (End)

pool-1-thread-1 (End)

pool-1-thread-5 (End)

all threads are finished