Java threading – The Executor framework

June 4, 2009 | by Rafał Borowiec

The main goal of this article is to present the better mechanism of executing threads in Java and to show that working directly with threads is no longer necessary. Thanks to the Executor framework, working with threads in Java is easier. And there are people who claim that working with threads in Java is easier than in most other languages.

Defining threads in Java

Multithreading is increasingly becoming the part of every modern application these days. The program is multithreaded if it is able to run more than one thread at the same time.

In Java, defining threads is straightforward and it is done either by extending the java.lang.Thread class and overriding its run() method or, more common, by implementing the java.lang.Runnable interface.

Let's look at the example of defining a thread by implementing the Runnable interface:

```
1.  class SomeRunnable implements Runnable {
2.      @Override
3.      public void run() {
4.          for (int i = 0; i < 3; i++) {
5.              System.out.println(Thread.currentThread().getName() + " counter="
6.                      + i);
7.          }
8.      }
9.  }
```

To have the above code run by a separate thread the new java.lang.Thread object must be instantiated and started as it is shown below:

```
1.  public class Runner {
2.      public static void main(String[] args) {
3.          // instantiate a runnable object
4.          Runnable runnable = new SomeRunnable();
5.          // instantiate 2 threads and pass the runnable object
6.          Thread thread_0 = new Thread(runnable);
7.          Thread thread_1 = new Thread(runnable);
8.          // start the threads
9.          thread_0.start();
10.         thread_1.start();
11.     }
12. }
```

Using the Runnable interface is better than extending the Thread class. One of the main advantages of doing that is the ability of using an **Executor** framework instead of directly using the Thread class to create and execute new threads: constructing new threads is relatively expensive because it involves interaction with the operating system.

The Executor framework

The Executor framework is a framework for standardizing invocation, scheduling, execution, and control of asynchronous tasks according to a set of execution policies.

The Executor Framework was introduced in Java 1.5 and it is a part of java.util.concurrent package. The main interface for the framework is Executor interface and its subinterfaces: **ExecutorService** and **ScheduledExecutorService**. Please refer to Executor API [ExecutorAPI] to see the detailed description of the interfaces.

Let's take a look at the simple example of using the **Executor** framework to run previously created task using fixed thread pool:

```
1.  public class Runner {
2.      public static void main(String[] args) {
3.          Runnable runnable = new SomeRunnable();
4.          // create a fixed thread pool
5.          ExecutorService pool = Executors.newFixedThreadPool(3);
6.          // run the task 5 times using the pool
7.          for (int i = 0; i < 5; i++) {
8.              pool.execute(runnable);
9.          }
10.         pool.shutdown();
11.     }
12. }
```

In the above example I used the ExecutorService interface which is an Executor that provides a number of useful methods to control the task execution. For creation of the concrete ExecutorService I used the Executors class and one of its static factory methods: newFixedThreadPool(). Please refer to Executor API [ExecutorAPI].

Task scheduling

The ScheduledExecutorService is the Executor that supports scheduling of tasks based on the relative time or to run task periodically. Example of scheduling a task to run after a 1000 ms:

```
1.  public class Runner {
2.      public static void main(String[] args) {
3.          Runnable runnable = new SomeRunnable();
4.
5.          ScheduledExecutorService thread = Executors
6.                  .newSingleThreadScheduledExecutor();
7.
8.          thread.schedule(runnable, 1000, TimeUnit.MILLISECONDS);
```

```
9.          thread.shutdown();
10.     }
11. }
```

Summary

The article barely scratched the surface of the Executor framework. Thus, I encourage the reader to study the framework's documentation to see its power and simplicity.

References and useful links

1. [ExecutorAPI] – The API documentation of the framework is comprehensive and should be a good starting point to get along with the framework. Examples included – http://java.sun.com/javase/6/docs/api/java/util/concurrent/Executor.html
2. "Concurrency: Executors" article at http://www.javalobby.org/forums/thread.jspa?threadID=16252&tstart=0
3. "The Executor Framework" from "Java Concurrency in Practice" at http://book.javanb.com/java-concurrency-in-Practice/ch06lev1sec2.html