1. Future is just interface. Behind the scene, the implementation of Future is FutureTask.
2. You can absolutely use FutureTask manually but you will lose the advantages of using Executor(pooling thread, limit the thread, etc).
3. Using FutureTask is quite similar to using the old Thread and the start() method.

**public** **class** FutureTaskTest **implements** Callable {

@Override

**public** String call() **throws** Exception {

System.***out***.println("Inside call " + Thread.*currentThread*().getName());

Thread.*sleep*(1000);

**return** "success";

}

**public** **static** **void** main(String[] args) **throws** InterruptedException, ExecutionException {

FutureTaskTest f = **new** FutureTaskTest();

FutureTask ft = **new** FutureTask(f);

ft.run();

System.***out***.println(ft.get());

}

}

**How to cancel a task or thread that is running?**

A task can be configured to cancel when a timeout occurs. But the task needs to be cancelled manually to prevent running further.

Future<?> task = **executorService**.submit(runnable);

**try** {

task.get(5, TimeUnit.Seconds); //5 Seconds

} **catch** (TimeoutException e) {

System.***err***.println(**"Timeout occurred."**);

task.cancel(**true**);

}