Executors, ExecutorService, newScheduledThreadPool

2020

Executors

Зачем?

В Java без `Executors` для каждой задачи пришлось бы создавать новые потоки.

Как запускается поток без`Executors`:

```
new Thread (aRunnableObject).start ();
```

Как запускается поток с `Executors`:

```
Executor executor = some Executor factory method;
exector.execute(aRunnable);
```

Executors

Преимущество

- Абстрагируемся от низкоуровневых деталей управления потоками.

Executors

Недостаток

- Абстрагируемся от низкоуровневых деталей управления потоками.

Executors

Когда НЕ использовать

- Приложению требуется точное количество потоков и бизнес логика - простая.
- Требуется простая мномопоточная модель, без пула потоков.
- Вы уверены, что управляете жизненным циклом потоков +
    - обработка исключений
    - взаимодействия между потоками

Executors

?

```
ExecutorService exec = Executors.newFixedThreadPool( nThreads: 10);

for(int i = 0; i < 10; i++) {
    exec.submit(new Task(i));
}

for(int i = 0; i < 10; i++) {
    exec.submit(new Task(i));
}
exec.shutdown();
```

```
List<Thread> threads = new ArrayList<Thread>();
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(new Task(i));
    threads.add(t);
    t.start();
}

for(Thread t: threads) {
    try {
        t.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

threads = new ArrayList<Thread>();
for(int i = 0; i < 10; i++) {
    Thread t = new Thread(new Task(i));
    threads.add(t);
    t.start();
}

for(Thread t: threads) {
    try {
        t.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

ExecutorService

FAQ

В основу ExecutorService положен интерфейс Executor, в котором определен один метод :

```
void execute(Runnable thread);
```

ExecutorService

FAQ

При вызове метода **execute** исполняется поток **thread**. То есть, метод **execute** запускает указанный поток на исполнение.

Как запускается поток без`Executors`:

```
new Thread (aRunnableObject).start ();
```

Как запускается поток с `Executors`:

```
Executor executor = some Executor factory method;
exector.execute(aRunnable);
```

ExecutorService

FAQ

Интерфейс ExecutorService расширяет свойства Executor, дополняя его методами управления исполнением и контроля. Так в интерфейс ExecutorService включен метод shutdown(), позволяющий останавливать все потоки исполнения, находящиеся под управлением экземпляра ExecutorService. Также в интерфейсе ExecutorService определяются методы, которые запускают потоки исполнения FutureTask, возвращающие результаты и позволяющие определять статус остановки.

ExecutorService

FAQ

ExecutorService исполняет асинхронный код в одном или нескольких потоках. Создание инстанса ExecutorService'а делается либо вручную через конкретные имплементации (ScheduledThreadPoolExecutor или ThreadPoolExecutor), но проще будет использовать фабрики класса Executors. Например, если надо создать пул с 2мя потоками, то делается это так:

```
ExecutorService service = Executors.newFixedThreadPool(2);
```

ExecutorService

Методы

Метод	**Описание**
boolean **awaitTermination**(long timeout, TimeUnit unit)	Блокировка до тех пор, пока все задачи не завершат выполнение после запроса на завершение работы или пока не наступит тайм-аут или не будет прерван текущий поток, в зависимости от того, что произойдет раньше
List<Future<T>> **invokeAll** (Collection<? extends Callable<T>> tasks)	Выполнение задач с возвращением списка задач с их статусом и результатами завершения
List<Future<T>> **invokeAll** (Collection<? extends Callable<T>> tasks, long timeout, TimeUnit unit)	Выполнение задач с возвращением списка задач с их статусом и результатами завершения в течение заданного времени
T **invokeAny**(Collection<? extends Callable<T>> tasks)	Выполнение задач с возвращением результата успешно выполненной задачи (т. е. без создания исключения), если таковые имеются
T **invokeAny**(Collection<? extends Callable<T>> tasks, long timeout, TimeUnit unit)	Выполнение задач в течение заданного времени с возвращением результата успешно выполненной задачи (т. е. без создания исключения), если таковые имеются
boolean **isShutdown()**	Возвращает true, если исполнитель сервиса остановлен (shutdown)
boolean **isTerminated()**	Возвращает true, если все задачи исполнителя сервиса завершены по команде остановки (shutdown)
void **shutdown()**	Упорядоченное завершение работы, при котором ранее отправленные задачи выполняются, а новые задачи не принимаются
List<Runnable> **shutdownNow()**	Остановка всех активно выполняемых задач, остановка обработки ожидающих задач, возвращение списка задач, ожидающих выполнения
Future<T> **submit**(Callable<T> task)	Завершение выполнения задачи, возвращающей результат в виде объекта Future
Future<?> **submit**(Runnable task)	Завершение выполнения задачи, возвращающей объект Future, представляющий данную задачу
Future<T> **submit**(Runnable task, T result)	Завершение выполнения задачи, возвращающей объект Future, представляющий данную задачу

newScheduledThreadPool

FAQ

Иногда требуется выполнение кода асихронно и периодически или требуется выполнить код через некоторое время, тогда на помощь приходит ScheduledExecutorService. Он позволяет поставить код выполняться в одном или нескольких потоках и сконфигурировать интервал или время, на которое выполненение будет отложено. Интервалом может быть время между двумя последовательными запусками или время между окончанием одного выполнения и началом другого.

newScheduledThreadPool

FAQ

Например, если требуется отложить выполнение на 5 секунд, потребуется следующий код:

```
ScheduledExecutorService service = Executors.newScheduledThreadPool();
service.schedule(new Runnable() { ... }, 5, TimeUnit.SECONDS);
```

newScheduledThreadPool

FAQ

Если требуется назначить выполнение каждую секунду:

```
ScheduledExecutorService service = Executors.newScheduledThreadPool();
service.scheduleAtFixedRate(new Runnable() { ... }, 0, 1, TimeUnit.SECONDS);
```

newScheduledThreadPool

FAQ

Если требуется назначить выполнение кода с промежутком 1 секунда между выполнениями:

```
ScheduledExecutorService service = Executors.newScheduledThreadPool();
service.scheduleWithFixedDelay(new Runnable() { ... }, 0, 1, TimeUnit.SECONDS);
```

newScheduledThreadPool

FAQ

Метод создает пул потоков, который планирует запуск с учетом:

- Задержки
- Периодическое выполнение

Принимает в качестве параметра - количество потоков в пуле.

Возвращает: ScheduledExecutorService - пул запланированных потоков.