LESSON 11

Main.java

```java
try (var ex = Executors
  .newVirtualThreadPerTaskExecutor()) {
  ex.submit(task);
}
```

Java

Virtual Threads

Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

JAVA PROGRAMMING

LESSON 11

Concurrency and Virtual Threads

Threads, ExecutorService, synchronization, CompletableFuture and virtual threads

Threads

Threads.java

```java
Thread t = new Thread(() -> {
    System.out.println("working in " + Thread.currentThread().getName());
});
t.start();
t.join();               // wait for it
```

- A thread is a separate path of execution. Start one with a Runnable lambda, and join waits …
- Creating raw threads by hand is rare today. Executors manage them for you.

01

ExecutorService

Pool.java

```
try (ExecutorService pool = Executors.newFixedThreadPool(4)) {
    Future<Integer> f = pool.submit(() -> countLines("big.log"));
    System.out.println(f.get());       // waits for the result
}
```

- An ExecutorService runs tasks on a pool of threads. submit returns a Future, and get waits …
- Use it in a try-with-resources block, and it shuts down cleanly when you are done.

02

Shared State and Synchronization



- When threads change shared data, updates can be lost. synchronized lets only one thread …
- Better still, use thread-safe tools: AtomicInteger for counters and ConcurrentHashMap for …

03

CompletableFuture

Async.java

```
var price = CompletableFuture.supplyAsync(() -> fetchPrice("book"));
var rate  = CompletableFuture.supplyAsync(() -> fetchRate("AFN"));

price.thenCombine(rate, (p, r) -> p * r)
     .thenAccept(total -> System.out.println("Total: " + total))
     .exceptionally(e -> { System.err.println(e); return null; })
     .join();
```

- CompletableFuture chains asynchronous steps: supplyAsync starts work, thenApply transforms …
- exceptionally handles failures, all without blocking a thread while waiting.

04

Virtual Threads

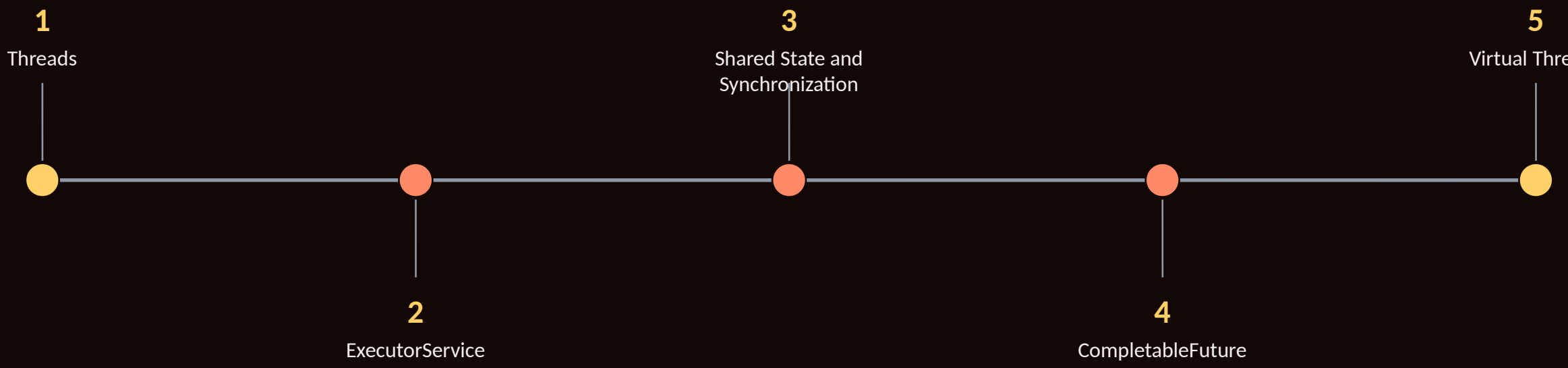
Virtual.java

```java
try (var exec = Executors.newVirtualThreadPerTaskExecutor()) {
    for (int i = 0; i < 100_000; i++) {
        int id = i;
        exec.submit(() -> {
            Thread.sleep(Duration.ofSeconds(1));
            return id;
        });
    }
}   // 100,000 tasks finish in about one second
```

- Java 21 added virtual threads: extremely lightweight threads managed by the JVM. You can …
- Write simple blocking code, one thread per task, and still handle huge numbers of network …

05

Lesson Map


1
Threads
2
ExecutorService
3
Shared State and Synchronization
4
CompletableFuture
5
Virtual Threads

Sources and Further Reading





Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari