Spring Boot + @Async = 王炸！！

老炮说Java 2022-09-30 14:00 发表于山西

收录于合集
#老炮说Java 378 #Spring Boot 23

老炮说Java

十年老炮程序员带你玩技术

公众号

异步调用几乎是处理高并发Web应用性能问题的万金油，那么什么是“异步调用”？

“异步调用”对应的是“同步调用”，同步调用指程序按照定义顺序依次执行，每一行程序都必须等待上一行程序执行完成之后才能执行；异步调用指程序在顺序执行时，不等待异步调用的语句返回结果就执行后面的程序。

同步调用

下面通过一个简单示例来直观的理解什么是同步调用：

定义Task类，创建三个处理函数分别模拟三个执行任务的操作，操作消耗时间随机取（10秒内）

```
@Component
public class Task {

    public static Random random =new Random();

    public void doTaskOne() throws Exception {
        System.out.println("开始做任务一");
```

```
            long  start = System.currentTimeMillis();
            Thread.sleep(random.nextInt(10000));
            long  end = System.currentTimeMillis();
            System.out.println("完成任务一，耗时：" + (end - start) + "毫秒");
        }

        public void doTaskTwo() throws Exception {
            System.out.println("开始做任务二");
            long start = System.currentTimeMillis();
            Thread.sleep(random.nextInt(10000));
            long end = System.currentTimeMillis();
            System.out.println("完成任务二，耗时：" + (end - start) + "毫秒");
        }

        public void doTaskThree() throws Exception {
            System.out.println("开始做任务三");
            long start = System.currentTimeMillis();
            Thread.sleep(random.nextInt(10000));
            long end = System.currentTimeMillis();
            System.out.println("完成任务三，耗时：" + (end - start) + "毫秒");
        }

}
```

在单元测试用例中，注入Task对象，并在测试用例中执行doTaskOne、doTaskTwo、doTaskThree三个函数。

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = Application.class)
public class ApplicationTests {

    @Autowired
    private Task task;

    @Test
    public void test() throws Exception {
        task.doTaskOne();
        task.doTaskTwo();
        task.doTaskThree();
    }

}
```

执行单元测试，可以看到类似如下输出：

```
开始做任务一
完成任务一，耗时：4256毫秒

开始做任务二
完成任务二，耗时：4957毫秒

开始做任务三
完成任务三，耗时：7173毫秒
```

任务一、任务二、任务三顺序的执行完了，换言之doTaskOne、doTaskTwo、doTaskThree三个函数顺序的执行完成。

异步调用

上述的同步调用虽然顺利的执行完了三个任务，但是可以看到执行时间比较长，若这三个任务本身之间不存在依赖关系，可以并发执行的话，同步调用在执行效率方面就比较差，可以考虑通过异步调用的方式来并发执行。

在Spring Boot中，我们只需要通过使用@Async注解就能简单的将原来的同步函数变为异步函数，Task类改在为如下模式：

```
@Component
public class Task {

    @Async
    public void doTaskOne() throws Exception {
        // 同上内容，省略
    }

    @Async
    public void doTaskTwo() throws Exception {
        // 同上内容，省略
    }

    @Async
    public void doTaskThree() throws Exception {
        // 同上内容，省略
    }
```

```
}
```

为了让@Async注解能够生效，还需要在Spring Boot的主程序中配置@EnableAsync，如下所示：

```
@SpringBootApplication
@EnableAsync
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

此时可以反复执行单元测试，您可能会遇到各种不同的结果，比如：

- 没有任何任务相关的输出
- 有部分任务相关的输出
- 乱序的任务相关的输出

原因是目前doTaskOne、doTaskTwo、doTaskThree三个函数的时候已经是异步执行了。

主程序在异步调用之后，主程序并不会理会这三个函数是否执行完成了，由于没有其他需要执行的内容，所以程序就自动结束了，导致了不完整或是没有输出任务相关内容的情况。

> 注：@Async所修饰的函数不要定义为static类型，这样异步调用不会生效

异步回调

为了让doTaskOne、doTaskTwo、doTaskThree能正常结束，假设我们需要统计一下三个任务并发执行共耗时多少，这就需要等到上述三个函数都完成调动之后记录时间，并计算结果。

那么我们如何判断上述三个异步调用是否已经执行完成呢？我们需要使用Future来返回异步调用的结果，就像如下方式改造doTaskOne函数：

```
@Async
public Future<String> doTaskOne() throws Exception {
    System.out.println("开始做任务一");
    long start = System.currentTimeMillis();
    Thread.sleep(random.nextInt(10000));
    long end = System.currentTimeMillis();
    System.out.println("完成任务一，耗时：" + (end - start) + "毫秒");
    return new AsyncResult<>("任务一完成");
}
```

按照如上方式改造一下其他两个异步函数之后，下面我们改造一下测试用例，让测试在等待完成三个异步调用之后来做一些其他事情。

```
@Test
public void test() throws Exception {

    long start = System.currentTimeMillis();

    Future<String> task1 = task.doTaskOne();
    Future<String> task2 = task.doTaskTwo();
    Future<String> task3 = task.doTaskThree();

    while(true) {
        if(task1.isDone() && task2.isDone() && task3.isDone()) {
            // 三个任务都调用完成，退出循环等待
            break;
        }

        Thread.sleep(1000);

    }

    long end = System.currentTimeMillis();

    System.out.println("任务全部完成，总耗时：" + (end - start) + "毫秒");
}
```

看看我们做了哪些改变：

- 在测试用例一开始记录开始时间
- 在调用三个异步函数的时候，返回Future类型的结果对象

- 在调用完三个异步函数之后，开启一个循环，根据返回的Future对象来判断三个异步函数是否都结束了。若都结束，就结束循环；若没有都结束，就等1秒后再判断。

跳出循环之后，根据结束时间 - 开始时间，计算出三个任务并发执行的总耗时。

执行一下上述的单元测试，可以看到如下结果：

```
开始做任务一
开始做任务二
开始做任务三

完成任务三，耗时：37毫秒
完成任务二，耗时：3661毫秒
完成任务一，耗时：7149毫秒

任务全部完成，总耗时：8025毫秒
```

可以看到，通过异步调用，让任务一、二、三并发执行，有效的减少了程序的总运行时间。

> 原文：developer.aliyun.com/article/694020

收录于合集 #Spring Boot 23

上一篇
太强了，一个注解搞定接口返回数据脱敏

下一篇
SpringBoot 实现 Excel 自由导入导出，性能强的离谱，用起来还特优雅！

阅读原文

喜欢此内容的人还喜欢

推荐12个值得学习的TypeScript宝库！

前端充电宝

Ubuntu20.04+docker+jenkins+飞书实现自动化发布

趣编程ACE

Spring Boot 实现万能文件在线预览，已开源，真香！！

开源前线