17、Spring Boot与任务

2020年7月7日 17:19

- 1、异步任务

  在Java应用中，绝大多数情况下都是通过同步的方式来实现交互处理的；但是在处理与第三方系统交互的时候，容易造成响应迟缓的情况，之前大部分都是使用多线程来完成此类任务，其实，在Spring 3.x之后，就已经内置了@Async来完美解决这个问题。

  两个注解：

  @EnableAysnc、@Aysnc

  发送邮件或者处理数据等不想阻塞下边的任务，可以使用异步任务

  - 同步：当我们执行某个功能时，在没有得到结果之前，这个调用就不能返回！简单点就是说必须 等前一件事做完才能做下一件事；
  - 异步：和同步则是相对的，当我们执行某个功能后，我们并不需要立即得到结果，我们额可以正常地 做其他操作，这个功能可以在完成后通知或者回调来告诉我们；比如应用正在运行， 发现新版本了，想后台更新，这个时候一般我们会开辟出一条后台线程，用于下载新版本的apk，但是这个时候 我们还可以使用应用中的其他功能！ 我们执行下载功能后，我们就无需去关心它的下载过程，当下载完毕后通知我们就可以了~
  - springboot使用异步

  1) 开启异步

  ```
  package com.lhq.task;
  import org.springframework.boot.SpringApplication;
  import org.springframework.boot.autoconfigure.SpringBootApplication;
  import org.springframework.scheduling.annotation.EnableAsync;
  @EnableAsync//开启异步
  @SpringBootApplication
  public class Springboot04TaskApplication {
      public static void main(String[] args) {
          SpringApplication.run(Springboot04TaskApplication.class,
  args);
      }
  }
  ```

  2) 使用异步注解

  ```
  package com.lhq.task.service;
  import org.springframework.scheduling.annotation.Async;
  import org.springframework.stereotype.Service;
  @Service//加入容器
  public class AsyncService {
      //告诉spring这是异步方法
  ```

```
    @Async
    public void hello(){
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("处理数据中...");
    }
}
```

- 2、定时任务

  项目开发中经常需要执行一些定时任务，比如需要在每天凌晨时候，分析一次前一天的日志信息。Spring为我们提供了异步执行任务调度的方式，提供 TaskExecutor 、TaskScheduler 接口。

  两个注解：@EnableScheduling、@Scheduled

  cron表达式：

字段	允许值	允许的特殊字符
秒	0-59	, - * /
分	0-59	, - * /
小时	0-23	, - * /
日期	1-31	, - * ? / L W C
月份	1-12	, - * /
星期	0-7或SUN-SAT 0,7是SUN	, - * ? / L C #

特殊字符	代表含义
,	枚举
-	区间
*	任意
/	步长
?	日/星期冲突匹配
L	最后
W	工作日
C	和calendar联系后计算过的值
#	星期，4#2，第2个星期四

  - springboot使用定时任务

  1）开启定时任务注解

```
package com.lhq.task;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.annotation.EnableScheduling;

@EnableAsync//开启异步
@EnableScheduling//开启定时任务注解
@SpringBootApplication
public class Springboot04TaskApplication {
    public static void main(String[] args) {
        SpringApplication.run(Springboot04TaskApplication.class,
args);
    }
}
```

2) 使用异步@Scheduled注解

```
package com.lhq.task.service;

import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;

@Service
public class ScheduledService {
    /*
* second（秒）, minute（分）, hour（时）, day of month（日）,
month（月）,day of week（周几）
* @code "0 * * * * MON-FRI" 周一到周五每一分钟启动一次 0代表每个0秒启
动，*代表所有时刻
*  【0 0/5 14,18 * * ?】 每天14点整，和18点整，每隔5分钟执行一次
*  【0 15 10 ? * 1-6】 每个月的周一至周六10:15分执行一次
*  【0 0 2 ? * 6L】每个月的最后一个周六凌晨2点执行一次
*  【0 0 2 LW * ?】每个月的最后一个工作日凌晨2点执行一次
*  【0 0 2-4 ? * 1#1】每个月的第一个周一凌晨2点到4点期间，每个整点都执
行一次;
* */
    //@Scheduled(cron = "0 * * * * MON-FRI")
    //@Scheduled(cron = "0,1,2,3,4 * * * * MON-FRI")
    //@Scheduled(cron = "0-4 * * * * MON-FRI")
    @Scheduled(cron = "0/4 * * * * MON-FRI")//每4秒执行一次
    public void hello(){
        System.out.println("hello ... ");
```

```
        }
    }
```