狂神说SpringBoot15：异步、定时、邮件任务

秦疆 狂神说 2020-03-26

前言

在我们的工作中，常常会用到异步处理任务，比如我们在网站上发送邮件，后台会去发送邮件，此时前台会造成响应不动，直到邮件发送完毕，响应才会成功，所以我们一般会采用多线程的方式去处理这些任务。还有一些定时任务，比如需要在每天凌晨的时候，分析一次前一天的日志信息。还有就是邮件的发送，微信的前身也是邮件服务呢？这些东西都是怎么实现的呢？其实SpringBoot都给我们提供了对应的支持，我们上手使用十分的简单，只需要开启一些注解支持，配置一些配置文件即可！那我们来看看吧~

最后编辑于2020.3.26 作者：狂神说

异步任务

1、创建一个service包

2、创建一个类AsyncService

异步处理还是非常常用的，比如我们在网站上发送邮件，后台会去发送邮件，此时前台会造成响应不动，直到邮件发送完毕，响应才会成功，所以我们一般会采用多线程的方式去处理这些任务。

编写方法，假装正在处理数据，使用线程设置一些延时，模拟同步等待的情况；

```
@Service
public class AsyncService {

   public void hello(){
       try {
           Thread.sleep(3000);
       } catch (InterruptedException e) {
           e.printStackTrace();
       }
       System.out.println("业务进行中....");
   }
}
```

3、编写controller包

4、编写AsyncController类

我们去写一个Controller测试一下

```
@RestController
public class AsyncController {

   @Autowired
   AsyncService asyncService;

   @GetMapping("/hello")
   public String hello(){
       asyncService.hello();
       return "success";
  }

}
```

5、访问http://localhost:8080/hello进行测试，3秒后出现success，这是同步等待的情况。

问题：我们如果想让用户直接得到消息，就在后台使用多线程的方式进行处理即可，但是每次都需要自己手动去编写多线程的实现的话，太麻烦了，我们只需要用一个简单的办法，在我们的方法上加一个简单的注解即可，如下：

6、给hello方法添加@Async注解；

```
//告诉Spring这是一个异步方法
@Async
public void hello(){
   try {
       Thread.sleep(3000);
  } catch (InterruptedException e) {
       e.printStackTrace();
  }
   System.out.println("业务进行中....");
}
```

SpringBoot就会自己开一个线程池，进行调用！但是要让这个注解生效，我们还需要在主程序上添加一个注解@EnableAsync ，开启异步注解功能；

```
@EnableAsync //开启异步注解功能
@SpringBootApplication
public class SpringbootTaskApplication {

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

}
```

7、重启测试，网页瞬间响应，后台代码依旧执行！

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的时候，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，提供了两个接口。

- TaskExecutor接口
- 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个星期三

测试步骤：

1、创建一个ScheduledService

我们里面存在一个hello方法，他需要定时执行，怎么处理呢？

```
@Service
public class ScheduledService {

   //秒   分   时     日   月   周几
   //0 * * * * MON-FRI
```

```
    //注意cron表达式的用法;
    @Scheduled(cron = "0 * * * * 0-7")
    public void hello(){
        System.out.println("hello.....");
    }
}
```

2、这里写完定时任务之后，我们需要在主程序上增加@EnableScheduling 开启定时任务功能

```
@EnableAsync //开启异步注解功能
@EnableScheduling //开启基于注解的定时任务
@SpringBootApplication
public class SpringbootTaskApplication {

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

}
```

3、我们来详细了解下cron表达式;

http://www.bejson.com/othertools/cron/

4、常用的表达式

```
(1) 0/2 * * * * ?   表示每2秒 执行任务
(1) 0 0/2 * * * ?   表示每2分钟 执行任务
(1) 0 0 2 1 * ?   表示在每月的1日的凌晨2点调整任务
(2) 0 15 10 ? * MON-FRI   表示周一到周五每天上午10:15执行作业
(3) 0 15 10 ? 6L 2002-2006   表示2002-2006年的每个月的最后一个星期五上午10:15执行作
(4) 0 0 10,14,16 * * ?   每天上午10点，下午2点，4点
(5) 0 0/30 9-17 * * ?   朝九晚五工作时间内每半小时
(6) 0 0 12 ? * WED   表示每个星期三中午12点
(7) 0 0 12 * * ?   每天中午12点触发
(8) 0 15 10 ? * *   每天上午10:15触发
(9) 0 15 10 * * ?    每天上午10:15触发
(10) 0 15 10 * * ?   每天上午10:15触发
(11) 0 15 10 * * ? 2005   2005年的每天上午10:15触发
(12) 0 * 14 * * ?    在每天下午2点到下午2:59期间的每1分钟触发
(13) 0 0/5 14 * * ?   在每天下午2点到下午2:55期间的每5分钟触发
(14) 0 0/5 14,18 * * ?    在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发
(15) 0 0-5 14 * * ?   在每天下午2点到下午2:05期间的每1分钟触发
(16) 0 10,44 14 ? 3 WED   每年三月的星期三的下午2:10和2:44触发
(17) 0 15 10 ? * MON-FRI   周一至周五的上午10:15触发
(18) 0 15 10 15 * ?   每月15日上午10:15触发
(19) 0 15 10 L * ?   每月最后一日的上午10:15触发
(20) 0 15 10 ? * 6L   每月的最后一个星期五上午10:15触发
(21) 0 15 10 ? * 6L 2002-2005   2002年至2005年的每月的最后一个星期五上午10:15触发
(22) 0 15 10 ? * 6#3   每月的第三个星期五上午10:15触发
```

邮件任务

邮件发送，在我们的日常开发中，也非常的多，Springboot也帮我们做了支持

- 邮件发送需要引入spring-boot-start-mail
- SpringBoot 自动配置MailSenderAutoConfiguration
- 定义MailProperties内容，配置在application.yml中
- 自动装配JavaMailSender
- 测试邮件发送

测试：

1、引入pom依赖

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

看它引入的依赖，可以看到 jakarta.mail

```
<dependency>
   <groupId>com.sun.mail</groupId>
   <artifactId>jakarta.mail</artifactId>
   <version>1.6.4</version>
   <scope>compile</scope>
</dependency>
```

2、查看自动配置类：MailSenderAutoConfiguration

这个类中存在bean，JavaMailSenderImpl

```
)
@ConditionalOnJndi
class MailSenderJndiConfiguration {
    private final MailProperties properties;

    MailSenderJndiConfiguration(MailProperties properties) {
        this.properties = properties;
    }

    @Bean
    public JavaMailSenderImpl mailSender(Session session) {
        JavaMailSenderImpl sender = new JavaMailSenderImpl();
        sender.setDefaultEncoding(this.properties.getDefaultEncoding().name());
        sender.setSession(session);
        return sender;
    }
```


然后我们去看下配置文件

```
@ConfigurationProperties(
   prefix = "spring.mail"
)
public class MailProperties {
   private static final Charset DEFAULT_CHARSET;
   private String host;
   private Integer port;
   private String username;
   private String password;
   private String protocol = "smtp";
   private Charset defaultEncoding;
   private Map<String, String> properties;
   private String jndiName;
}
```

3、配置文件：

```
spring.mail.username=24736743@qq.com
spring.mail.password=你的qq授权码
spring.mail.host=smtp.qq.com
# qq需要配置ssl
spring.mail.properties.mail.smtp.ssl.enable=true
```

获取授权码：在QQ邮箱中的设置->账户->开启pop3和smtp服务

4、Spring单元测试

```
@Autowired
JavaMailSenderImpl mailSender;

@Test
public void contextLoads() {
    //邮件设置1：一个简单的邮件
    SimpleMailMessage message = new SimpleMailMessage();
    message.setSubject("通知-明天来狂神这听课");
    message.setText("今晚7:30开会");

    message.setTo("24736743@qq.com");
    message.setFrom("24736743@qq.com");
    mailSender.send(message);
}

@Test
public void contextLoads2() throws MessagingException {
    //邮件设置2：一个复杂的邮件
    MimeMessage mimeMessage = mailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, true);

    helper.setSubject("通知-明天来狂神这听课");
    helper.setText("<b style='color:red'>今天 7:30来开会</b>",true);

    //发送附件
    helper.addAttachment("1.jpg",new File(""));
    helper.addAttachment("2.jpg",new File(""));

    helper.setTo("24736743@qq.com");
```

```
    helper.setFrom("247367430@qq.com");

    mailSender.send(mimeMessage);
}
```

查看邮箱，邮件接收成功！

我们只需要使用Thymeleaf进行前后端结合即可开发自己网站邮件收发功能了！

狂神讲解的配套视频地址 https://www.bilibili.com/video/BV1PE411i7CV

end

视频同步更新，这次一定！