# ERP项目笔记 Day-09

## 库存预警

1. 库存预警报表
   1. 需求及实现思路
      1. 需求

统计每种商品的库存数量与待发货数量，如果库存数量小于待发货数量则要进行库存预警



* + 1. 实现思路

将整个个查询结果拆成2个子查询，再合并起来



* 1. SQL语句编写

这种带子查询的语句相对复杂，我们分步来实现

1. 先实现当前商品库存统计，查询出每种商品的库存数量



结果如下：

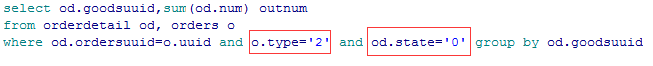


这种结果只能查询存在库存的商品数量，而库存中不存在的商品则不会显示出来。我们可以通过左外连方式来进行查询。使用nvl方法把null转成0

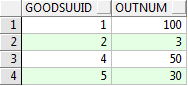




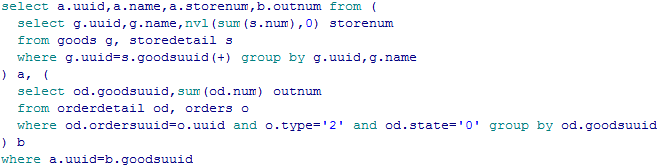
1. 实现销售订单中商品数量的统计，统计每种待发货的商品数量



结果如下：



1. 通过商品编号将上面这2种查询组合起来



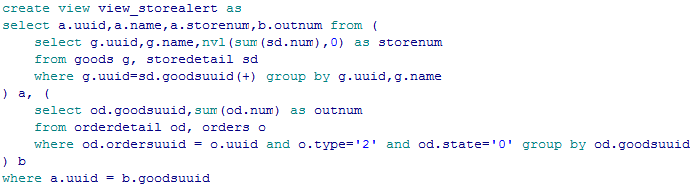
结果如下：



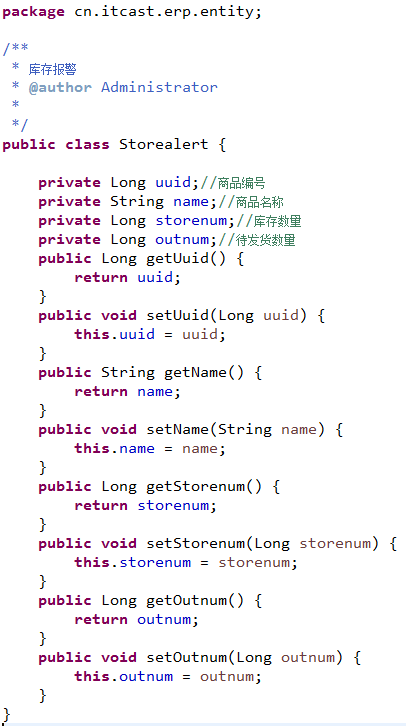
* 1. 视图创建

我们上边的语句的查询结果是我们经常要用的，每次都写很繁琐，所以我们可以把常用的复杂查询直接建立为视图，把它当成hibernate中的一个表来处理，这样可以极大简化我们的开发，便于日常维护。

创建视图sql语句:

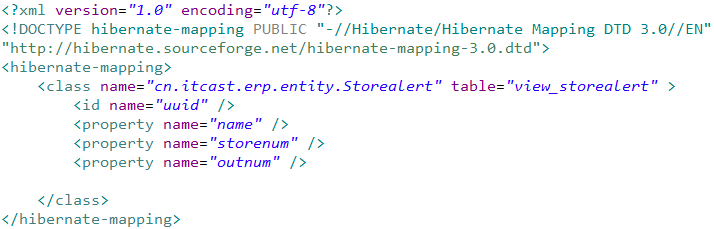


* 1. 代码编写
     1. 创建实体类



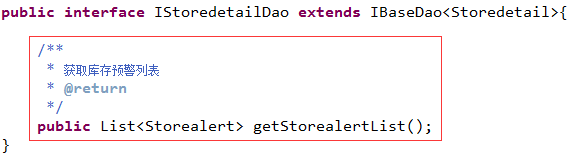
* + 1. 创建映射文件

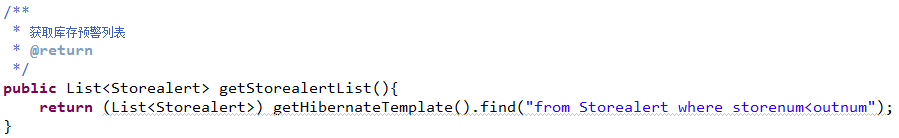
映射文件storealert.hbm.xml



* + 1. 数据访问层实现

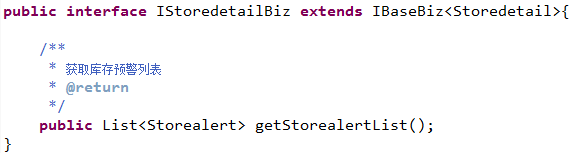
在IStoredetailDao和StoredetailDao中加入方法与实现

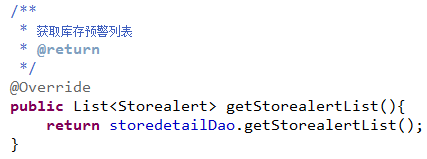




* + 1. 业务逻辑层实现

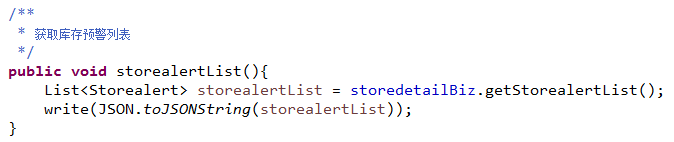
在IStoredetailBiz和StoredetailBiz中添加方法与实现





* + 1. Action层实现

在StoredetailAction中加入方法：



* + 1. 页面代码

1. 添加js/storealert.js文件



1. 添加storealert.html文件



1. 运行结果

http://localhost:8080/erp/storealert.html



1. JavaMail发送预警邮件（扩展）
   1. 需求分析

实现发送邮件预警功能，提醒相关人员进行采购。点击“发送警报邮件”按钮即可发送库存报警。

如果存在报警商品并成功发送，提示发送成功

如果不存在报警商品，提示没有库存报警商品



* 1. JaveMail简介

JavaMail，顾名思义，提供给开发者处理电子邮件相关的编程接口。它是Sun发布的用来处理email的API。它可以方便地执行一些常用的邮件传输。JavaMail API是Sun公司为方便Java开发人员在应用程序中实现邮件发送和接收功能而提供的一套标准开发包，它支持一些常用的邮件协议，如：**SMTP、POP3、IMAP**。开发人员使用JavaMail API编写邮件处理软件时，无须考虑邮件协议底层的实现细节，只要调用JavaMail开发包中相应的API类就可以了。

使用 JavaMail，我们可以实现类似 OutLook、FoxMail 的软件。

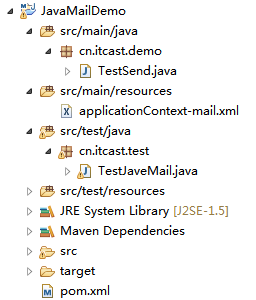
**Spring**的**javaMailSender**对JavaMail进行封装，简化了开发，告别繁琐的API

* 1. 入门Demo
     1. 邮箱设置

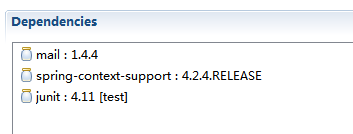
登陆邮箱后，进入设置，开启POP3/SMTP



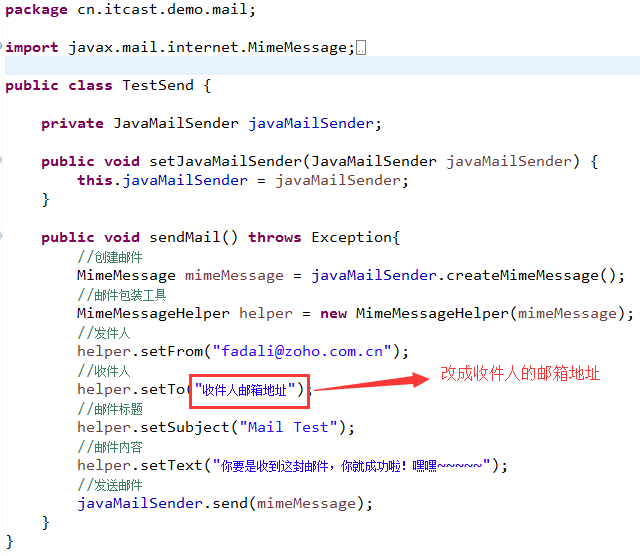
* + 1. 创建工程



* + 1. 添加依赖



* + 1. 编写发送Email类



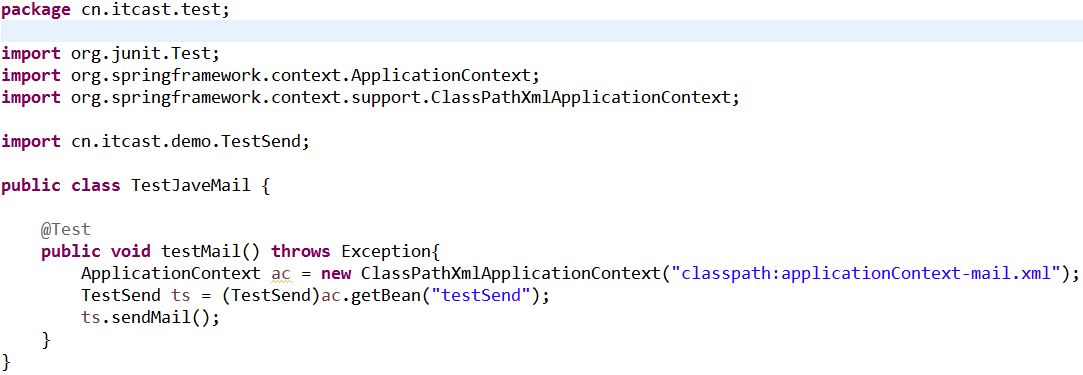
* + 1. 编写spring配置文件

创建applicationContext\_mail.xml文件，copy文件头约束后，添加下面代码

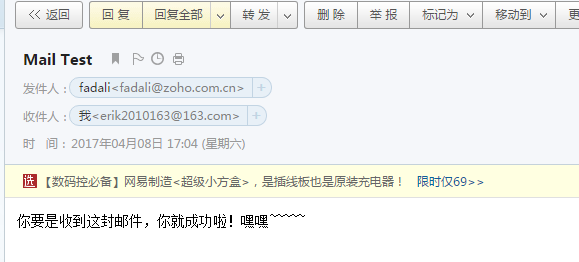


* + 1. 编写测试代码

创建测试类，并运行测试

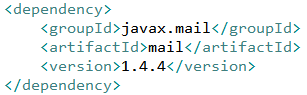


结果：



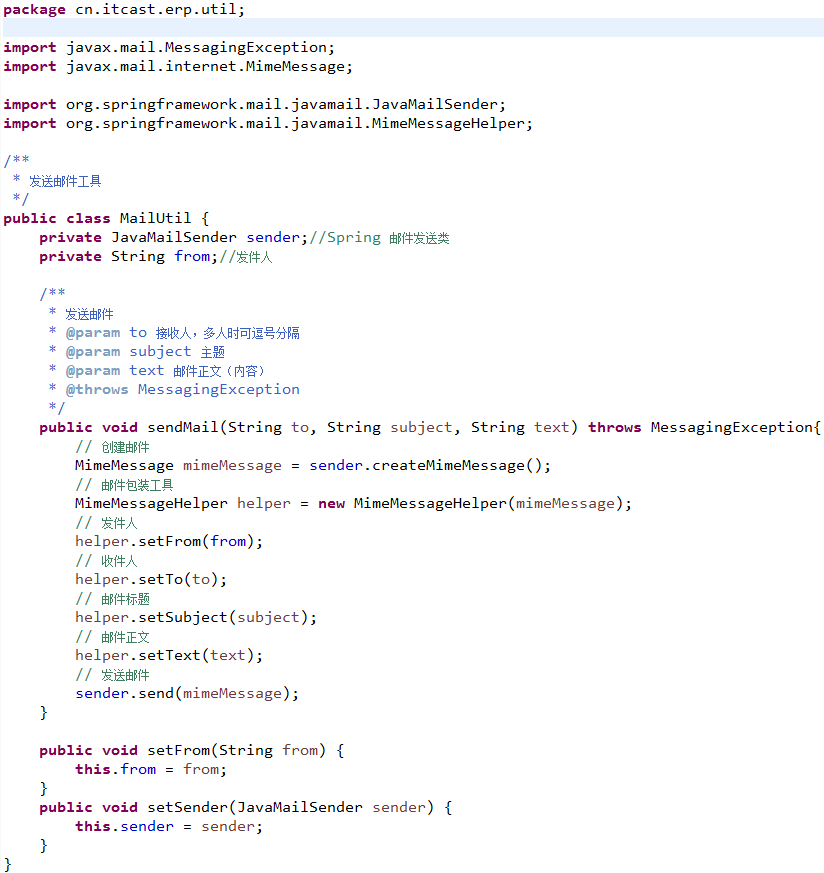
* 1. ERP中实现发送预警邮件
     1. 添加依赖

在父工程的pom.xml中添加java mail的依赖



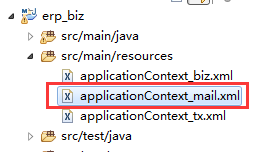
* + 1. 编写发送邮件工具类

在业务逻辑层中创建MailUtil



* + 1. 编写spring配置文件

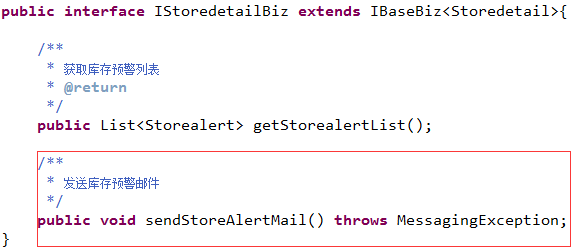
创建applicationContext\_mail.xml



|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd*  *"*>  <!-- 邮件发送器 -->  <bean id=*"javaMailSender"* class=*"org.springframework.mail.javamail.JavaMailSenderImpl"*>  <!-- 服务器 -->  <property name=*"host"* value=*"smtp.zoho.com.cn"*></property>  <!-- 端口 -->  <property name=*"port"* value=*"465"*></property>  <!-- 发送邮件协议 -->  <property name=*"protocol"* value=*"smtp"*></property>  <!-- 登陆用户名 -->  <property name=*"username"* value=*"fadali@zoho.com.cn"*></property>  <!-- 密码 -->  <property name=*"password"* value=*"bB123456"*></property>    <property name=*"javaMailProperties"*>  <props>  <!-- 需要验证发件人 -->  <prop key=*"mail.smtps.auth"*>true</prop>  <!-- 开启SSL协议加密码 -->  <prop key=*"mail.smtp.ssl.enable"*>true</prop>  </props>  </property>  </bean>  <!-- 邮件工具类 -->  <bean id=*"mailUtil"* class=*"cn.itcast.erp.util.MailUtil"*>  <property name=*"sender"* ref=*"javaMailSender"*></property>  <property name=*"from"* value=*"fadali@zoho.com.cn"*></property>  </bean>  </beans> |

* + 1. 编写业务逻辑类

给IStoredetailBiz和StoredetailBiz添加sendStoreAlertMail方法





* + 1. 修改applicationContext\_biz.xml

storedetailBiz节点，添加：



* + 1. 编写action代码

修改StoredetailAction，添加代码：



* + 1. 编写js代码

修改storealert.js:



1. Quartz定时发送预警邮件（重点）
   1. 需求及实现思路

定时查询库存预警信息，一旦存在库存预警的商品，则发邮件通知相关人员

* 1. Quartz 框架

Quartz是OpenSymphony开源组织在Job scheduling领域又一个开源项目，它完全由 Java 写成，并设计用于 J2SE 和 J2EE 应用中。Quartz可以用来创建简单或为运行十个，百个，甚至是好几万个Jobs这样复杂的程序。它提供了巨大的灵活性而不牺牲简单性。你能够用它来为执行一个作业而创建简单的或复杂的调度。

1. Job

表示一个任务（工作），要执行的具体内容。

1. JobDetail

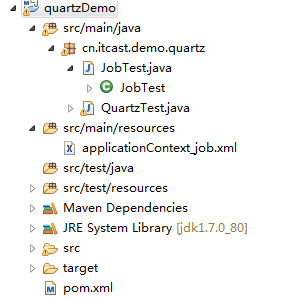
JobDetail 表示一个具体的可执行的调度程序，Job 是这个可执行程调度程序所要 执行的内容，另外 JobDetail 还包含了这个任务调度的方案和策略。

告诉调度容器，将来执行哪个类(job)的哪个方法

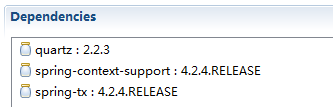
1. Trigger 是一个类，代表一个调度参数的配置，描述触发Job执行的时间触发规则。一个Job可以对应多个Trigger，但一个Trigger只能对应一个Job
2. Scheduler 代表一个调度容器，一个调度容器中可以注册多个 JobDetail 和Trigger。 Scheduler可以将Trigger绑定到某一JobDetail中，这样当Trigger触发时，对应的Job就被执行。

Note: 当JobDetail和Trigger在scheduler容器上注册后，形成了装配好的作业（JobDetail和Trigger所组成的一对儿），就可以伴随容器启动而调度执行了。

* 1. 入门Demo
     1. 创建Demo工程，加入依赖

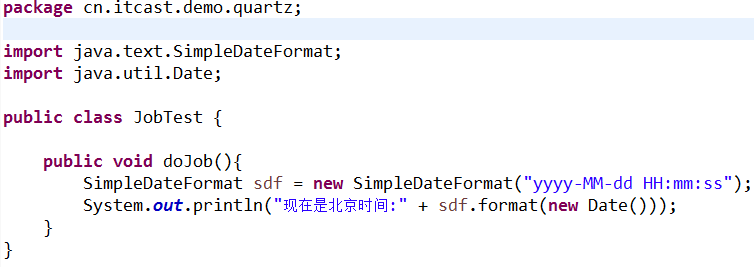


添加依赖:



* + 1. 编写任务类

创建JobTest类，代码如下



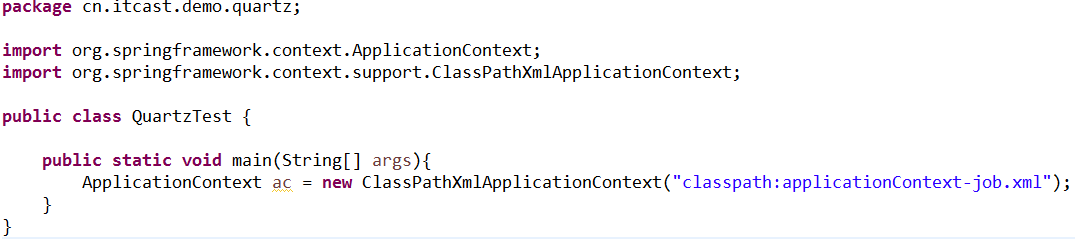
* + 1. 配置文件

添加applicationContext-job.xml文件

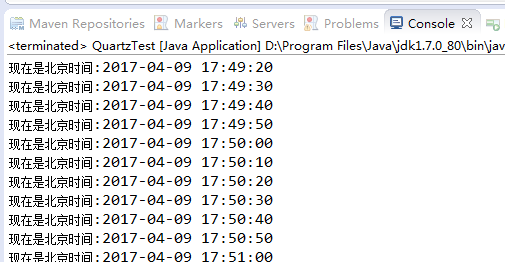


* + 1. 测试

创建测试类QuartzTest，注：不能使用junit做测试，因为一运行完，程序就结束了。



测试结果:



* 1. 表达式cron详解

Cron 表达式是一个字符串，字符串以 5 或 6 个空格隔开，分为 6 或 7 个域，每一个域代表一个含义。也叫七子表达式。

* + 1. 域

1. Seconds（秒）：可出现", - \* /"四个字符，有效范围为 0-59 的整数
2. Minutes（分钟）：可出现", - \* /"四个字符，有效范围为 0-59 的整数
3. Hours（小时）:可出现", - \* /"四个字符，有效范围为 0-23 的整数
4. DayofMonth（日 of 月）:可出现", - \* / ? LW C"八个字符，有效范围为 1-31 的整数
5. Month（月）:可出现", - \* /"四个字符，有效范围为 1-12 的整数
6. DayofWeek（日 of 星期）:可出现", - \* / ? L C #"八个字符，有效范围为 1-7 的整数

1 表示星期天，2 表示星期一， 依次类推

1. Year（年）:可出现", - \* /"四个字符，有效范围为 1970-2099 年
   * 1. 字符含义
2. **\*** 表示匹配该域的任意值，假如在 Minutes 域使用\*, 即表示每分钟都会触发事件。
3. **?** 表示不指定值。只能用在 DayofMonth 和 DayofWeek 两个域。因为DayofMonth 和 DayofWeek 会相互影响。例如想在每月的 20 日触发调度，不管20 日到底是星期几，则只能使用如下写法： 13 13 15 20 \* ?, 其中最后一位只能用？，而不能使用\*，如果使用\*表示不管星期几都会触发。
4. **-** 表示范围，例如在 Minutes 域使用 5-20，表示从 5 分到 20 分钟每分钟触发一次
5. **/** 表示起始时间开始触发，然后每隔固定时间触发一次，例如在 Minutes 域使用 5/20,则意味着 5 分钟触发一次，而 25，45 等分别触发一次
6. **,** 表示列出枚举值值。例如：在 Minutes 域使用 5,20，则意味着在 5 和 20 分每分钟触发一次。
7. **L** 表示最后，只能出现在 DayofMonth 和 DayofWeek 域。如果在 DayofMonth写 L 表示这个月的最后一天，如果在 DayofWeek 写 L 表示每个星期的最后一天（星期六） 。如果在 DayofWeek 域使用 5L,意味着在最后的一个星期四触发。
8. **W** 表示最近有效工作日(周一到周五),只能出现在 DayofMonth 域，系统将在离指定日期的最近的有效工作日触发事件。例如：在 DayofMonth 使用 5W，如果 5 日是星期六，则将在最近的工作日：星期五，即 4 日触发。如果 5 日是星期天，则在 6 日(周一)触发；如果 5 日在星期一到星期五中的一天，则就在 5 日触发。另外一点，W 的最近寻找不会跨过月份
9. **LW**:这两个字符可以连用，表示在某个月最后一个工作日，即最后一个星期五
10. **#**:用于确定每个月第几个星期几，只能出现在 DayofWeek 域。例如在 4#2，表示某月的第二个星期三。
    * 1. 示例

"0 0 12 \* \* ?" 每天中午 12 点触发

"0 15 10 ? \* \*" 每天上午 10:15 触发

"0 15 10 \* \* ?" 每天上午 10:15 触发

"0 15 10 \* \* ? \*" 每天上午 10:15 触发

"0 15 10 \* \* ? 2005" 2005 年的每天上午 10:15 触发

"0 \* 14 \* \* ?" 在每天下午 2 点到下午 2:59 期间的每 1 分钟触发

"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

"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

"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

"0 10,44 14 ? 3 4" 每年三月的星期三的下午 2:10 和 2:44 触发

"0 15 10 ? \* MON-FRI" 周一至周五的上午 10:15 触发

"0 15 10 15 \* ?" 每月 15 日上午 10:15 触发

"0 15 10 L \* ?" 每月最后一日的上午 10:15 触发

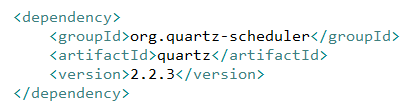
"0 15 10 ? \* 6L" 每月的最后一个星期五上午 10:15 触发

"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

"0 15 10 ? \* 6#3" 每月的第三个星期五上午 10:15 触发

* 1. 代码实现
     1. 添加quartz依赖

打开父工程的pom.xml，添加quartz依赖：



* + 1. 创建任务类

在业务逻辑层中新建MailJob类，包名为cn.itcast.erp.job

|  |
| --- |
| **package** cn.itcast.erp.job;  **import** java.util.List;  **import** javax.mail.MessagingException;  **import** cn.itcast.erp.biz.IStoredetailBiz;  **import** cn.itcast.erp.entity.Storealert;  /\*\*  \* 后台定时检测库存预警  \* 如果存在库存预警，则发预警邮件给相关工作人员  \* **@author** Administrator  \*  \*/  **public** **class** MailJob {  /\*\*  \* 商品库存业务  \*/  **private** IStoredetailBiz storedetailBiz;  **public** **void** sendStorealertMail(){  //查询是否存在库存预警  List<Storealert> storealertList = storedetailBiz.getStorealertList();  **if**(storealertList.size() > 0){  **try** {  //调用 业务发送预警邮件  storedetailBiz.sendStoreAlertMail();  } **catch** (MessagingException e) {  e.printStackTrace();  }  }  }  **public** **void** setStoredetailBiz(IStoredetailBiz storedetailBiz) {  **this**.storedetailBiz = storedetailBiz;  }  } |

* + 1. 配置文件

创建applicationContext\_job.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd*  *"*>  <!-- 定义一个任务类 -->  <bean id=*"job"* class=*"cn.itcast.erp.job.MailJob"*>  </bean>  <!-- 任务类描述 -->  <bean id=*"jobDetail"* class=*"org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"*>  <property name=*"targetObject"* ref=*"job"*></property>  <property name=*"targetMethod"* value=*"sendStorealertMail"*></property>  </bean>  <!-- 触发器 -->  <bean id=*"mailTrigger"* class=*"org.springframework.scheduling.quartz.CronTriggerFactoryBean"*>  <property name=*"jobDetail"* ref=*"jobDetail"*></property>  <!--触发时间cron表达式, 七子表达式. 7个域 -->  <!-- 每天上午12:20分和下午16:20执行 -->  <property name=*"cronExpression"* value=*"0 20 12,16 \* \* ?"*></property>  </bean>  <!-- 任务调度管理容器 -->  <bean id=*"scheduler"* class=*"org.springframework.scheduling.quartz.SchedulerFactoryBean"* >  <property name=*"triggers"*>  <list>  <ref bean=*"mailTrigger"*/>  </list>  </property>  </bean>  </beans> |