[Quartz中时间表达式的设置-----corn表达式](http://blog.csdn.net/xufaxi/article/details/11805639)

2013-09-18 14:02 23645人阅读 [评论](http://blog.csdn.net/xufaxi/article/details/11805639#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/xufaxi/article/details/11805639#report)

 分类：



spring（2）  第三方工具、 插件的使用（15）



Quartz中时间表达式的设置-----corn表达式

时间格式: <!-- s m h d m w(?) y(?) -->,   分别对应: 秒>分>小时>日>月>周>年,

举例:

1.每天什么时候执行:

       <value>0 59 23 \* \* ?</value>: 如下为每天23:59:00开始执行

      <value>0 1,2,3 11,12 \* \* ? </value>: 每天11:01,11:02,11:03; 12:01,12:02,12:03分执行任务

2.每隔多久执行:

汇总说明:

Cron表达式的时间字段除允许设置数值外，还可使用一些特殊的字符，提供列表、范围、通配符等功能，细说如下：

●星号(\*)：可用在所有字段中，表示对应时间域的每一个时刻，例如，\*在分钟字段时，表示“每分钟”；

●问号（?）：该字符只在日期和星期字段中使用，它通常指定为“无意义的值”，相当于点位符；

●减号(-)：表达一个范围，如在小时字段中使用“10-12”，则表示从10到12点，即10,11,12；

●逗号(,)：表达一个列表值，如在星期字段中使用“MON,WED,FRI”，则表示星期一，星期三和星期五；

●斜杠(/)：x/y表达一个等步长序列，x为起始值，y为增量步长值。如在分钟字段中使用0/15，则表示为0,15,30和45秒，而5/15在分钟字段中表示5,20,35,50，你也可以使用\*/y，它等同于0/y；

==================================================

Quartz cron 表达式的格式十分类似于 UNIX cron 格式，但还是有少许明显的区别。区别之一就是 Quartz 的格式向下支持到秒级别的计划，而 UNIX cron 计划仅支持至分钟级。许多我们的触发计划要基于秒级递增的(例如，每45秒)，因此这是一个非常好的差异。  
  
在 UNIX cron 里，要执行的作业（或者说命令）是存放在 cron 表达式中的，在第六个域位置上。Quartz 用 cron 表达式存放执行计划。引用了 cron 表达式的CronTrigger 在计划的时间里会与 job 关联上。  
  
另一个与 UNIX cron 表达式的不同点是在表达式中支持域的数目。UNIX 给出五个域(分、时、日、月和周)，Quartz 提供七个域。表 5.1 列出了 Quartz cron 表达式支持的七个域。

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **表 5.1. Quartz Cron 表达式支持到七个域**  **名称** | **是否必须** | **允许值** | **特殊字符** |
| 秒 | 是 | 0-59 | , - \* / |
| 分 | 是 | 0-59 | , - \* / |
| 时 | 是 | 0-23 | , - \* / |
| 日 | 是 | 1-31 | , - \* ? / L W C |
| 月 | 是 | 1-12 或 JAN-DEC | , - \* / |
| 周 | 是 | 1-7 或 SUN-SAT | , - \* ? / L C # |
| 年 | 否 | 空 或 1970-2099 | , - \* / |

月份和星期的名称是不区分大小写的。FRI 和 fri 是一样的。  
  
域之间有空格分隔，这和 UNIX cron 一样。无可争辩的，我们能写的最简单的表达式看起来就是这个了：  
  
\* \* \* ? \* \*  
  
这个表达会每秒钟(每分种的、每小时的、每天的)激发一个部署的 job。  
  
**·理解特殊字符**  
  
同 UNIX cron 一样，Quartz cron 表达式支持用特殊字符来创建更为复杂的执行计划。然而，Quartz 在特殊字符的支持上比标准 UNIX cron 表达式更丰富了。  
  
**\* 星号**  
  
使用星号(\*) 指示着你想在这个域上包含所有合法的值。例如，在月份域上使用星号意味着每个月都会触发这个 trigger。  
  
表达式样例：  
  
0 \* 17 \* \* ?  
  
意义：每天从下午5点到下午5:59中的每分钟激发一次 trigger。它停在下午 5:59 是因为值 17 在小时域上，在下午 6 点时，小时变为 18 了，也就不再理会这个 trigger，直到下一天的下午5点。  
  
在你希望 trigger 在该域的所有有效值上被激发时使用 \* 字符。  
  
**? 问号**  
  
? 号只能用在日和周域上，但是不能在这两个域上同时使用。你可以认为? 字符是 "我并不关心在该域上是什么值。" 这不同于星号，星号是指示着该域上的每一个值。? 是说不为该域指定值。  
  
不能同时这两个域上指定值的理由是难以解释甚至是难以理解的。基本上，假定同时指定值的话，意义就会变得含混不清了：考虑一下，如果一个表达式在日域上有值11，同时在周域上指定了WED。那么是要 trigger 仅在每个月的11号，且正好又是星期三那天被激发？还是在每个星期三的11号被激发呢？要去除这种不明确性的办法就是不能同时在这两个域上指定值。  
  
只要记住，假如你为这两域的其中一个指定了值，那就必须在另一个字值上放一个 ?。  
  
表达式样例：  
  
0 10,44 14 ? 3 WEB  
  
意义：在三月中的每个星期三的下午 2:10 和 下午 2:44 被触发。  
  
**, 逗号**  
  
逗号 (,) 是用来在给某个域上指定一个值列表的。例如，使用值 0,15,30,45 在秒域上意味着每15秒触发一个 trigger。  
  
表达式样例：  
  
0 0,15,30,45 \* \* \* ?  
  
意义：每刻钟触发一次 trigger。  
  
**/ 斜杠**  
  
斜杠 (/) 是用于时间表的递增的。我们刚刚用了逗号来表示每15分钟的递增，但是我们也能写成这样0/15。  
  
表达式样例：  
  
0/15 0/30 \* \* \* ?  
  
意义：在整点和半点时每15秒触发 trigger。  
  
**- 中划线**  
  
中划线 (-) 用于指定一个范围。例如，在小时域上的 3-8 意味着 "3,4,5,6,7 和 8 点。" 域的值不允许回卷，所以像 50-10 这样的值是不允许的。  
  
表达式样例：  
  
0 45 3-8 ? \* \*  
  
意义：在上午的3点至上午的8点的45分时触发 trigger。  
  
**L 字母**  
L 说明了某域上允许的最后一个值。它仅被日和周域支持。当用在日域上，表示的是在月域上指定的月份的最后一天。例如，当月域上指定了JAN 时，在日域上的L会促使 trigger 在1月31号被触发。假如月域上是SEP，那么 L 会预示着在9月30号触发。换句话说，就是不管指定了哪个月，都是在相应月份的时最后一天触发 trigger。  
  
表达式 0 0 8 L \* ? 意义是在每个月最后一天的上午 8:00 触发 trigger。在月域上的 \* 说明是 "每个月"。  
  
当 L 字母用于周域上，指示着周的最后一天，就是星期六 (或者数字7)。所以如果你需要在每个月的最后一个星期六下午的 11:59 触发 trigger，你可以用这样的表达式0 59 23 ? \* L。  
  
当使用于周域上，你可以用一个数字与 L 连起来表示月份的最后一个星期 X。例如，表达式0 0 12 ? \* 2L 说的是在每个月的最后一个星期一触发 trigger。

|  |
| --- |
| **不要让范围和列表值与 L 连用**  虽然你能用星期数(1-7)与 L 连用，但是不允许你用一个范围值和列表值与 L 连用。这会产生不可预知的结果。 |

**W 字母**  
  
W 字符代表着平日 (Mon-Fri)，并且仅能用于日域中。它用来指定离指定日的最近的一个平日。大部分的商业处理都是基于工作周的，所以 W 字符可能是非常重要的。例如，日域中的15W 意味着 "离该月15号的最近一个平日。" 假如15号是星期六，那么 trigger 会在14号(星期五)触发，因为星期四比星期一（这个例子中是17号）离15号更近。（译者Unmi注：不会在17号触发的，如果是15W，可能会是在14号(15号是星期六)或者15号(15号是星期天)触发，也就是只能出现在邻近的一天，如果15号当天为平日直接就会当日执行）。W只能用在指定的日域为单天，不能是范围或列表值。  
  
**# 井号**  
  
# 字符仅能用于周域中。它用于指定月份中的第几周的哪一天。例如，如果你指定周域的值为6#3，它意思是某月的第三个周五 (6=星期五，#3意味着月份中的第三周)。另一个例子2#1意思是某月的第一个星期一 (2=星期一，#1意味着月份中的第一周)。注意，假如你指定#5，然而月份中没有第 5 周，那么该月不会触发。

此处的 Cron 表达式 cookbook 旨在为常用的执行需求提供方案。尽管不可能列举出所有的表达式，但下面的应该为满足你的业务需求提供了足够的例子。  
  
**·分钟的 Cron 表达式**

|  |  |
| --- | --- |
| **表 5.1. 包括了分钟频度的任务计划 Cron 表达式** | |
| **用法** | **表达式** |
| 每天的从 5:00 PM 至 5:59 PM 中的每分钟触发 | 0 \* 17 \* \* ? |
| 每天的从 11:00 PM 至 11:55 PM 中的每五分钟触发 | 0 0/5 23 \* \* ? |
| 每天的从 3:00 至 3:55 PM 和 6:00 PM 至 6:55 PM 之中的每五分钟触发 | 0 0/5 15,18 \* \* ? |
| 每天的从 5:00 AM 至 5:05 AM 中的每分钟触发 | 0 0-5 5 \* \* ? |

**·日上的 Cron 表达式**

|  |  |
| --- | --- |
| **表 5.2. 基于日的频度上任务计划的 Cron 表达式** | |
| **用法** | **表达式** |
| 每天的 3:00 AM | 0 0 3 \* \* ? |
| 每天的 3:00 AM (另一种写法) | 0 0 3 ? \* \* |
| 每天的 12:00 PM (中午) | 0 0 12 \* \* ? |
| 在 2005 中每天的 10:15 AM | 0 15 10 \* \* ? 2005 |

**·周和月的 Cron 表达式**

|  |  |
| --- | --- |
| **表 5.3. 基于周和/或月的频度上任务计划的 Cron 表达式** | |
| **用法** | **表达式** |
| 在每个周一,二, 三和周四的 10:15 AM | 0 15 10 ? \* MON-FRI |
| 每月15号的 10:15 AM | 0 15 10 15 \* ? |
| 每月最后一天的 10:15 AM | 0 15 10 L \* ? |
| 每月最后一个周五的 10:15 AM | 0 15 10 ? \* 6L |
| 在 2002, 2003, 2004, 和 2005 年中的每月最后一个周五的 10:15 AM | 0 15 10 ? \* 6L 2002-2005 |
| 每月第三个周五的 10:15 AM | 0 15 10 ? \* 6#3 |
| 每月从第一天算起每五天的 12:00 PM (中午) | 0 0 12 1/5 \* ? |
| 每一个 11 月 11 号的 11:11 AM | 0 11 11 11 11 ? |
| 三月份每个周三的 2:10 PM 和 2:44 PM | 0 10,44 14 ? 3 WED |

**八. 创建一个即刻触发的 Trigger**  
  
有时候，你需要立即执行一个 job。例如，想像一下，你正在构建一个 GUI 程序并允许用户能立刻执行。另一个例子，你或许已经检测到了某个 Job 未执行成功，因此你想要即刻重跑一次。在 Quartz 1.5，有几个方法被加入到了TriggerUtils 类中，使得实现那些事很容易了。代码 5.4 展示了如何部署一个 job，只让它立即执行一次。

顶

4

踩