在Web工程中实现任务计划调度 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022__E5_9C_A8Web_E5_B7_A5_E7_c104_144436.htm 好多朋友用过Windows的任务计划，也有不少程序迷自己曾写过时钟报警、系统自动关机等趣味程序，可却很少有朋友在Web工程中实现过类似功能。今天有空把笔者先前曾在Tomcat上实现的类似功能，搬出来与大家共享。 早在几年前，我公司跟某市财政局合作项目开发，为加强财政局对所属单位财务状况的有效监管，开发、实施了财政局数据中心项目。此项目采用B/S加C/S混合结构模式。财政局Web服务器上架设数据同步接收装置，由市属单位每天下班前把财务信息通过HTTP协议上传至财政局中心服务器，与Web服务器上的接收装置对接。财政局内部各部门需要查阅大量财务信息，获取完备的市属单位当前财务状况信息，各部门按职能划分，需要准确的获取各部门各自所关注的汇总信息，以财政报表的形式提供。 因财政数据量大，实时计算财政报表速度较慢，当初就考虑用报表缓存来减轻服务器的负担，但用缓存需要一个合理的缓存更新机制。考虑到各市属单位每天下班前才把财务数据上传，财政局每天所查看到的财务信息其实并不包括当天（除非有某位领导等到所属单位全部上传完之后才来查看信息，应该已经下班了），所以要是能实现任务计划调度，在每晚深夜把当天及历史财务信息汇总，更新缓存，速度瓶颈不就解决了吗。 当时由于系统核心是基于Web部署的，报表计算引擎也相应的部署在Tomcat容器上，因此如果想要借用Windows的任务计划来实现定时计算，就需要额外编写普通桌面应用程

序接口，稍显复杂。于是就琢磨着想在Web上实现，经过查阅较多相关资料，发现Java定时器（java.util.Timer）有定时触发计划任务的功能，通过配置定时器的间隔时间，在某一间隔时间段之后会自动有规律的调用预先所安排的计划任务（java.util.TimerTask）。另外，由于我们希望当Web工程启动时，定时器能自动开始计时，在整个Web工程的生命期里，定时器能在每晚深夜触发一次报表计算引擎。因此定时器的存放位置也值得考查，不能简单的存在于单个Servlet或JavaBean中，必须能让定时器宿主的存活期为整个Web工程生命期，在工程启动时能自动加载运行。结合这两点，跟Servlet上下文有关的侦听器就最合适不过了，通过在工程的配置文件中加以合理配置，会在工程启动时自动运行，并在整个工程生命期中处于监听状态。 下面就Servlet侦听器结合Java定时器来讲述整个实现过程。要运用Servlet侦听器需要实现javax.servlet.ServletContextListener接口，同时实现它的contextInitialized(ServletContextEvent event)和contextDestroyed(ServletContextEvent event)两个接口函数。考虑定时器有个建立和销毁的过程，看了前面两个接口函数，就不容置疑的把建立的过程置入contextInitialized，把销毁的过程置入contextDestroyed了。 我把ServletContextListener的实现类取名为ContextListener,在其内添加一个定时器，示例代码如下所示（为考虑篇幅，仅提供部分代码供读者参考）：

```
private java.util.Timer timer = null. public void contextInitialized(ServletContextEvent event) { timer = new java.util.Timer(true). event.getServletContext().log("定时器已启动"). timer.schedule(new MyTask(event.getServletContext()), 0,
```

60*60*1000). event.getServletContext().log("已经添加任务调度表"). } public void contextDestroyed(ServletContextEvent event) { timer.cancel(). event.getServletContext().log("定时器销毁"). } 以上代码中, timer.schedule(new MyTask(event.getServletContext()), 0, 60*60*1000)这一行为定时器调度语句，其中MyTask是自定义需要被调度的执行任务（在我的财政数据中心项目中就是报表计算引擎入口），从java.util.TimerTask继承，下面会重点讲述，第三个参数表示每小时(即60*60*1000毫秒)被触发一次，中间参数0表示无延迟。其它代码相当简单，不再详细说明。 下面介绍MyTask的实现，上面的代码中看到了在构造MyTask时，传入了javax.servlet.ServletContext类型参数,是为记录Servlet日志方便而传入，因此需要重载MyTask的构造函数（其父类java.util.TimerTask原构造函数是没有参数的）。在timer.schedule()的调度中,设置了每小时调度一次,因此如果想实现调度任务每24小时被执行一次，还需要判断一下时钟点，以常量C_SCHEDULE_HOUR表示(晚上12点，也即0点)。同时为防止24小时执行下来，任务还未执行完（当然，一般任务是没有这么长的），避免第二次又被调度以引起执行冲突，设置了当前是否正在执行的状态标志isRunning。示例代码如下所示： private static final int C_SCHEDULE_HOUR = 0. private static boolean isRunning = false. private ServletContext context = null. public MyTask(ServletContext context) { this.context = context. } public void run() { Calendar cal = Calendar.getInstance(). if (!isRunning) { if (C_SCHEDULE_HOUR == cal.get(Calendar.HOUR_OF_DAY)) { isRunning = true. context.log("开始执行指定任务"). //TODO 添

加自定义的详细任务，以下只是示例 int i = 0. while (i context.log("已完成任务的" i "/" 10). } isRunning = false. context.log("指定任务执行结束"). } } else { context.log("上一次任务执行还未结束"). } } 上面代码中“ //TODO…… ”之下四行是真正被调度执行的演示代码(在我的财政数据中心项目中就是报表计算过程)，您可以换成自己希望执行的语句。 到这儿，ServletContextListener和MyTask的代码都已完整了。最后一步就是把ServletContextListener部署到您的Web工程中去，在您工程的web.xml配置文件中加入如下三行： com.test.ContextListener 当然，上面的com.test得换成您自己的包名了。保存web.xml文件后，把工程打包部署到Tomcat中即可。任务会在每晚12点至凌晨1点之间被执行，上面的代码会在Tomcat的日志文件中记录如下：2003-12-05 0：21：39 开始执行指定任务2003-12-05 0：21：39 已完成任务的1/10 ……2003-12-05 0：21：39 已完成任务的10/102003-12-05 0：21：39 指定任务执行结束 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com