**淘宝**

2012

**EasySchedule使用介绍**

**白眉、柏林**

目录

[1 简介 1](#_Toc342492241)

[2 系统介绍 2](#_Toc342492242)

[2.1 系统特点 2](#_Toc342492243)

[2.2 技术架构 2](#_Toc342492244)

[2.3 功能简介 3](#_Toc342492245)

[2.4 任务执行流程 4](#_Toc342492246)

[2.5 使用介绍 4](#_Toc342492247)

[2.5.1 安装部署 4](#_Toc342492248)

[2.5.2 标准任务 5](#_Toc342492249)

[2.5.3 shell任务 11](#_Toc342492250)

[2.5.4 数据监控任务 13](#_Toc342492251)

[2.5.5 文件监控任务 14](#_Toc342492252)

[2.5.6 用户创建 15](#_Toc342492253)

[2.5.7 参数管理 15](#_Toc342492254)

[2.5.8 默认值说明 16](#_Toc342492255)

[2.5.9 系统参数说明 17](#_Toc342492256)

[2.5.10 告警管理 18](#_Toc342492257)

[2.5.11 异步任务 18](#_Toc342492258)

[2.5.12 报表管理 18](#_Toc342492259)

[2.5.13 其他模块 19](#_Toc342492260)

[3 表说明 19](#_Toc342492261)

[4 FAQ 21](#_Toc342492262)

# 简介

EasySchedule定时任务平台是一个以quartz开源框架为基础，整合WEBAPP作为操作平台，对常见需要按一定时间规则进行触发的任务进行部署和管理的控制平台。

# 系统介绍

## **系统特点**

1. Server和Client分别支持集群和分布式部署
2. 任务的执行与调度分离
3. 可视化管理所有任务
4. 任务状态持久化于DB
5. 完善的日志跟踪和告警策略
6. 任务支持异步调度
7. 灵活支持各种自定义任务，扩展方便

## **技术架构**

技术结构：



## **功能简介**

功能域：



## **任务执行流程**



## **使用介绍**

### 安装部署

搭建数据库

EasySchedule支持mysql数据库。

MySQL： 请执行es-mysql-ddl.txt，es-mysql-dml.txt中的语句。

由于EasySchedule数据层由JPA编写，故而你也很容易迁移到其他数据库。

部署应用

1. 下载安装tomcat
2. 项目根目录mvn install –Dmaven.test.skip=true打包
3. 配置tomcat context

<Context path=*"/es"* docBase=*"your\_es\_root\_path\easySchedule\es-server\target\easyschedule"* reloadable=*"true"* crossContext=*"true"*>

<easyscheduleDataSource配置>

</Context>

访问应用

HOSTS添加：127.0.01 es.taobao.net

访问：http://es.taobao.net:1212/es/index

帐号：admin

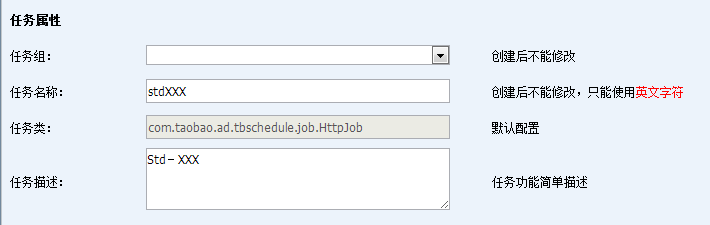
密码：aaa111

### 标准任务

下面我们通过创建一个任务并执行来大概了解整个系统的全貌。

#### 添加

**点击添加标准任务，首先填写任务属性**



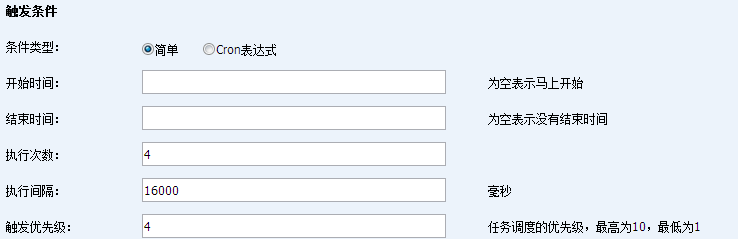
任务组一般指属于哪个应用，任务组的添加和配置参见参数管理。

**再填写任务参数**



每个任务参数的意义在右侧已经有很清晰的说明。默认可直接使用默认值，填写目标地址即可。

**填写触发条件**



触发条件支持简单按一定间隔时间执行的规则，也支持按不同时间频率的触发规则。

备注：简单类型的任务当执行次数为减为0的时候，该任务会被删除。

点击创建后，将在任务列表中出现该任务



可通过右侧做任务相关操作。

暂停：暂停该任务，EasySchedule将暂停该任务的调度。

立即执行：立即触发该任务。

修改：任务修改。

删除：任务删除。

详情：查看任务的详情。

历史：该任务历史记录，包含操作记录，执行记录，报警记录。

备注：所有操作都有相应的权限控制。

#### 接入

为了保证客户端的任务只被EasySchedule调度，需要在由于任务并非是只需要提供Http的URL即可，还需要在执行端做Token验证，保证只有ES能够调用到，所以需要对任务执行端进行一定程度的接入。

工程pom.xml增加依赖项（注意：es-client区分环境配置，开发和正式环境需要依赖不同的JAR包）  
开发测试等线下环境：

<dependency>

<groupId>com.taobao.ad</groupId>

<artifactId>es-client</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

正式环境:

<dependency>

<groupId>com.taobao.ad</groupId>

<artifactId>es-client</artifactId>

<version>1.0.0</version>

</dependency>

接入方式1：

通过标准request接受任务参数，用户编写SampleJobExecutor需要实现com.taobao.ad.easyschedule.executor.JobExecutor接口，任务执行完成后返回JobResult的JSON串至EasySchedule。action请求需要添加到登录及权限例外中；

struts2 Action示例代码:

1、编写struts-schdeule.xml文件

<action name=*"doXxxJob"* class=*"XxxScheduleAction"* method=*"doXxxJob"*>

<result name=*"success"* type=*"json"*>

<param name=*"root"*>result</param>

</result>

</action>

2、配置任务bean:

<bean id=*"sampleJobExecutor"* class=*"com.taobao.ad.easyschedule.executor.SampleJobExecutor"* />

3、编写XxxScheduleAction

**public** **class** XxxScheduleAction {

**private** SampleJobExecutor executor;

**private** JobResult result;

**public** String doXxxJob() {

HttpServletRequest request = ServletActionContext.getRequest();

JobResult result = HttpJobExecutor.*getInstance*().execute(request, executor);

**return** SUCCESS;

}

}

4、在SampleJobExecutor中编写任务代码

**package** com.taobao.ad.easyschedule.executor;

**import** java.io.IOException;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** com.taobao.ad.easyschedule.dataobject.JobData;

**import** com.taobao.ad.easyschedule.dataobject.JobResult;

/\*\*

\* 任务执行器(例子)

\*

\* **@author** baimei

\*

\*/

**public** **class** SampleJobExecutor **implements** JobExecutor {

**final** Logger logger = LoggerFactory.*getLogger*(SampleJobExecutor.**class**);

@Override

**public** JobResult execute(JobData jobData) **throws** IOException {

JobResult jobResult = JobResult.*succcessResult*();

**return** jobResult;

}

}

5、最后在任务平台添加标准任务

targetUrl填入当前任务的请求URL如：http://x.x.x.x/xxx/doXxxJob.action

其它配置选项详细参考任务添加说明；

接入方式2：

通过暴露Servlet接口方式接入

1、web.xml中增加

<servlet>

<servlet-name>ScheduleJobServlet</servlet-name>

<servlet-class>com.taobao.ad.easyschedule.servlet.ScheduleJobServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ScheduleJobServlet</servlet-name>

<url-pattern>/doScheduleJob</url-pattern>

</servlet-mapping>

2、排除web.xml各种filter的拦截，比如struts2中将默认filter作相应调整：

默认拦截配置：

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>

</filter>

修改成如下：

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

再在struts.xml中增加例外请求的正则配置

<constant name=*"struts.action.excludePattern"* value=*"/doScheduleJob.\*"*></constant>

注意：以上配置要求struts2在2.1.8或者以上版本。

3、配置任务bean:

<bean id=*"sampleJobExecutor"* class=*"com.taobao.ad.easyschedule.executor.SampleJobExecutor"* />

SampleJobExecutor需要实现com.taobao.ad. easyschedule.executor.JobExecutor接口；

4、在SampleJobExecutor中编写任务代码(参考接入方式1中第4步)；

5、在easySchedule中添加任务，任务地址：<http://x.x.x.x/xxx/doScheduleJob?beanId=sampleJobExecutor>

备注：如果任务返回token校验失败的信息则有两种可能。

1. agent和server的机器时间不一致。默认情况下，Token的有效期为30s。
2. agent和server的密钥不一致，导致生成MD5的token失败。

### shell任务

#### 添加

Shell任务配置的不同之处仅仅在于任务参数相关：



其中targetUrl的127.0.0.1配置Agent的地址。

#### 接入

管理调度Shell任务需要在目标机器上安装EasySchedule提供的Agent

目前es的Agent一共支持WebAgent,HttpAgent两种形式

WebAgent部署

1、安装tomcat

2、安装es-webagent.war包

3、Tomcat conf/Server.xml <Host>节点下增加

<Context path=*"/esagent"* docBase=*"your\_esagent\_root\_path/es-webagent"* debug=*"0"* reloadable=*"false"* crossContext=*"true"*>

</Context>

4、启动Tomcat容器，调用的URL为http//ip:port/esagent/doShellJob

HttpServerAgent部署

1、获取安装通讯模块的jar包 es-httpagent-1.0.1-SNAPSHOT.jar

2、启动JobHttpServer类

在/home/a/project/es-httpagent目录下 执行

java -jar /home/a/project/es-httpagent/es-httpagent-1.0.1.jar 9999 /esagent

说明：第一个参数代表发布的端口,第二个参数代表发布的context,默认使用9999 和/esagent

编写Shell注意事项

1. shell是否执行成功，是根据Shell的返回状态码确定。 使用exit 0或者exit 1等方式。
2. 当exit为0的时候，EasySchedule认为任务成功，当exit为其他状态码时，EasySchedule认为任务失败，并将状态码返回给服务端。

### 数据监控任务

#### 添加



如图所示：

1、targetUrl部署的是Agent，Agent的作用是连接数据库进行数据获取。

2、trackingSql是需要Agent执行的SQL，值得说明的是SQL仅仅支持返回单个值。

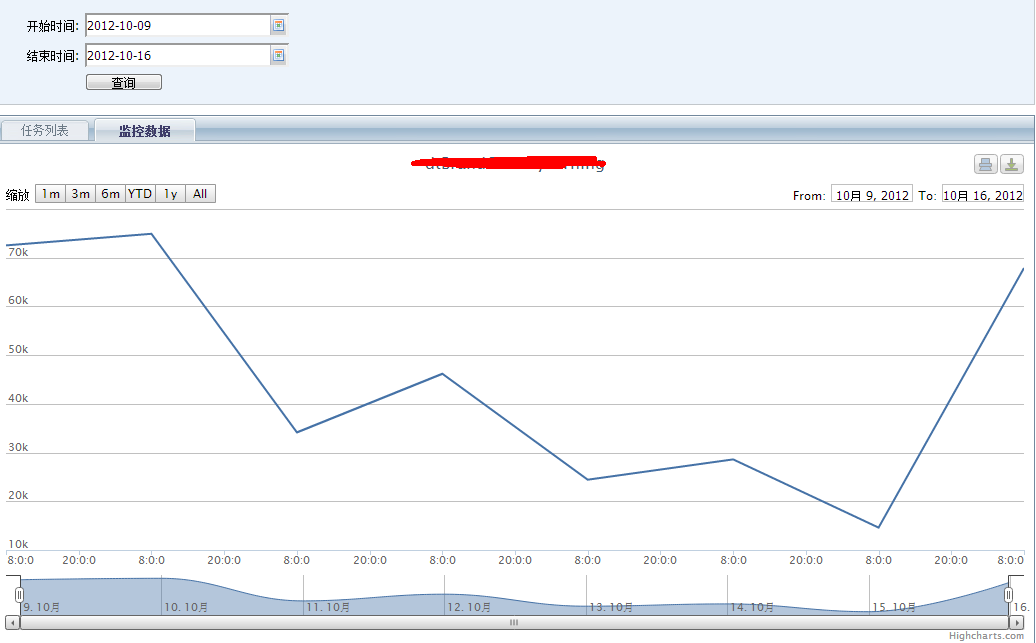
3、数据源是Agent中es-agent-ds.xml配置的数据源

4、数据源类型为ES-DS时获取Agent中es-agent-ds.xml的数据源，选择JNDI则Agent以JNDI的方式找数据源。

数据监控的任务生成后，会有独立的趋势图观看。



点击数据可查看监控值：

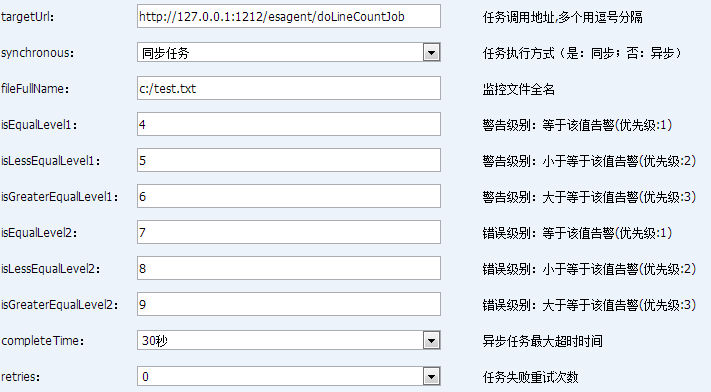


#### 接入

接入只需要安装Agent，在Agent中的es-agent-ds.xml文件中配置在任务中填写的数据源即可。

### 文件监控任务

#### 添加



1. targetUrl为Agent地址，需要替换IP，port和context。
2. fileFullName为需要监控的文件地址名称
3. 监控项和数据监控项作用类似。

#### 接入

安装Agent即可。

### 用户创建

点击配置管理—>用户管理—>用户添加

可创建用户，用户权限分两种，管理员和用户。 管理员可以做任何操作，用户只能做管理员授权组的操作。

权限分两种：

1. 数据权限

数据权限是针对任务组的，即管理员可以将任意组授权给指定用户。

1. 功能权限

功能权限一共数十项，管理员可以做任何操作。用户只能做管理员授权的组的任意操作。

用户只能做管理员授权的组的相关操作。管理员可以通过修改用户的任务组来实现组授权。

### 参数管理

参数管理主要针对全局的一些常量配置。目前列表参数如下：

|  |  |  |
| --- | --- | --- |
| CONFIGKEY | CONFIGVALUE | DESCRIPTION |
| INSTANCE\_CONTROL\_INTERFACE | http://#instance#:1212/es/ | 实例控制接口 |
| JOB\_DEFAULT\_CRONEXPRESSION | 0 0 8 \* \* ? | 任务默认参数：触发规则 |
| JOB\_DEFAULT\_NOTIFY\_FAILURE | 正在处理中 | 正在处理中 |
| JOB\_DEFAULT\_NOTIFY\_FINISHED\_MSG | 处理完成 | 处理完成 |
| JOB\_DEFAULT\_NOTIFY\_SUCCESS | 已处理完成 | 已处理完成 |
| JOB\_DEFAULT\_NOTIFY\_WAITING\_MSG | 处理中 | 处理中 |
| JOB\_DEFAULT\_PRIORITY | 4 | 任务默认参数：任务触发优先级 |
| JOB\_DEFAULT\_REPEATCOUNT | 4 | 任务默认参数：简单任务重复次数 |
| JOB\_DEFAULT\_REPEATINTERVAL | 16000 | 任务默认参数：简单任务重复间隔 |
| JOB\_DEFAULT\_TRIGGERTYPE | Cron | 任务默认参数：触发类型（Cron，Simple） |
| TBS\_TOP\_NOTIFY | True | 顶部通告开关 |
| TBS\_TOP\_NOTIFY\_MSG | 见SQL附件 | 顶部通告显示的值 |

备注：这里面的每一个参数基本都是必须的。特别是INSTANCE\_CONTROL\_INTERFACE参数的值需要用户修改为当前部署端口（替代1212）和当前部署的context（替代es），如果值不替换正确，无法对实例进行操作。

### 默认值说明

简单来理解，每一条默认值就代表某个任务中的一个参数控件或者某个全局的常量。目前共有5中任务类型：

CODEKEY：DATATRACKINGJOB\_JOBDATA 数据监控任务

CODEKEY：STOREDPROCEDURE\_JOBDATA 存储过程任务

CODEKEY：SHELLJOB\_JOBDATA SHELL任务

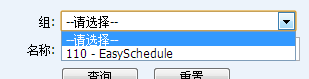
CODEKEY：LINECOUNTJOB\_JOBDATA 文件行数任务

CODEKEY：HTTPJOB\_JOBDATA HTTP标准任务

CODEKEY：JOB\_GROUP代表任务分组

JOB\_GROUP目前只有一个应用：





如果需要添加更多的应用可以在默认值中添加。

管理员通过修改用户的组，可以将不同的组权限授权给用户。

元素说明：

CODEKEY:类型

KEYCODE：控件名称

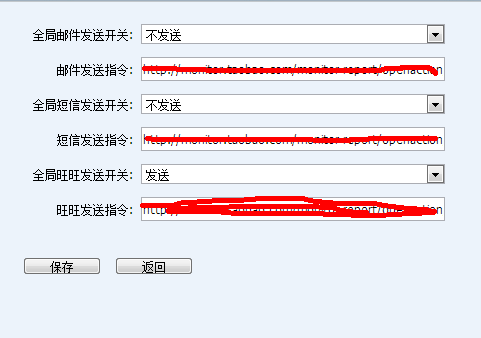
KEYNAME：控件默认值

KEYDESC：控件描述

SORTNUM：控件在不同的任务或者任务组的序列

### 系统参数说明

系统参数主要针对告警相关配置：



由于任务的成功或者失败会被一些用户关心，需要以旺旺、邮件、短信的方式发出告警。故而我们提供抽象的HTTP链接，固定的参数。让使用方实现报警具体逻辑，但是可以在系统参数中配置。

短信告警URL：

http://xxxx.xx.com/xx/xxx/alert.do?alertType=sms&smsList=#list#&subject=#subject#&msg=#content#

邮件告警URL：

http://xxxx.xx.com/xx/xxx/alert.do?alertType=email&emailList=#list#&subject=#subject#&msg=#content#

旺旺告警URL：

http://xxxx.xx.com/xx/xxx/alert.do?alertType=ww&wwList=#list#&subject=#subject#&msg=#content#

#list#中多个人告警以逗号为分割符号。#subject#指发送的标题，#content#指发送的内存，EasySchedule中的报警标题和内容都已经固化。

告警的内容示例：

成功：应用名称+ 任务名称 + "\_成功\_"+ 任务返回的信息

失败：应用名称+ 任务名称 + "\_失败\_"+ 任务返回的信息

### 告警管理

告警管理—告警信息订阅：



由于不同的人关心不同的系统和不同系统的状态返回。 故而有告警信息订阅管理，另外。当多个订阅者同时收到短信但是不知道谁在处理，和是否处理完成的情况下。可以点击**我正在处理中**，和**我已经处理完成**通知其他订阅者，让其他订阅者也能够实时的了解任务失败或成功的处理状态进度。

当一个比较重要的任务失败后用户可以选择是否需要多次通知任务告警接收人。

该重复告警扫描通过以下URL触发实现：

<http://ip:port/es/checkRepeatAlarm>

某个任务的重复告警次数是在任务中配置该选项对应数值即可：



如果任务失败后一定时间内没有操作人员点击**我正在处理中**或者**我已经处理完成**，任务会持续重复报警，直至报警次数增减至0后结束。

### 异步任务

EasySchedule用以下的方式保证异步任务可控。

1. 配置异步任务完成时间。

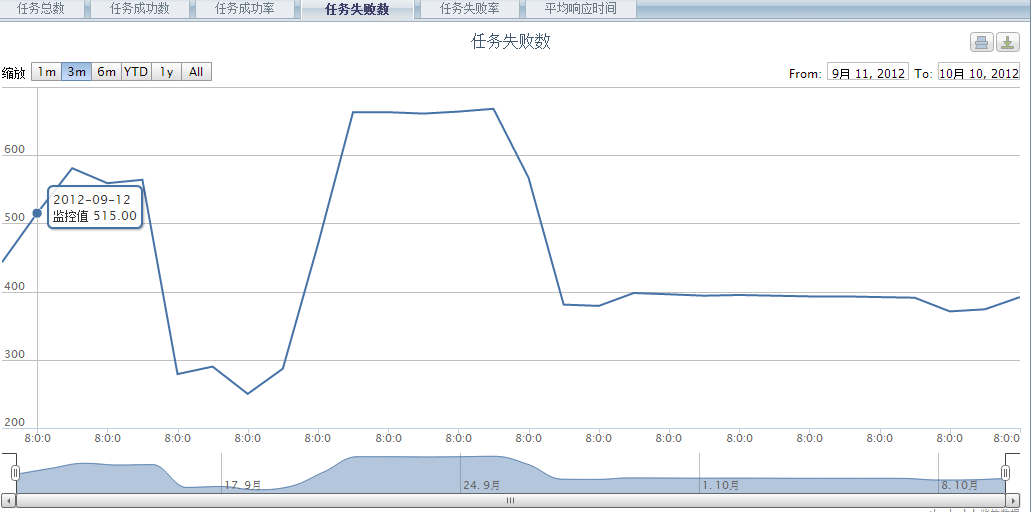


当一个异步任务被触发，还没有完成回调的时候会进入异步任务队列。

1. 配置<http://ip:port/es/checkAsyncJobQueue>,作用是检查异步任务队列有没有在规定的时候内回调，如果在规定的时间内没有回调则认为任务失败。
2. 异步队列的信息可以查看任务管理中的异步任务队列列表。

### 报表管理

EasySchedule对任务的执行情况作了不同维度的统计。主要包括任务总数，任务成功数，任务成功率，任务失败数，任务失败率，平均响应时间等；



图：任务统计信息

用户也可根据相关的表信息做邮件的报表发送等功能的衍生和扩展。

如果需要生成天维度的报表，需要在ES个配置一个默认的任务。URL：

<http://127.0.0.1:9999/es/initJobReport> 该URL每天初始化该天的天报表。

用户也可根据我们的报表相关数据，将这些数据以邮件的方式透明化出来发送给所有人。

### 其他模块

调试工具

调试工具帮助用户在线上或者线下做生成token等调试。

实例管理

es可以部署多台机器，在实例管理中可以统一维护各个实例。

异步任务队列

某些执行时间较长的任务会使用异步执行的方式，异步任务队列列出当前正在执行没有回调的任务列表。

# 表说明

|  |  |
| --- | --- |
| **表名称** | **说明** |
| es\_calendars | 以Blob 类型存储 Quartz 的 Calendar 信息 |
| es\_cron\_triggers | 存储 Cron Trigger，包括 Cron 表达式和时区信息 |
| es\_fired\_triggers | 存储与已触发的 Trigger 相关的状态信息，以及相联 Job 的执行信息 |
| es\_paused\_trigger\_grps | 存储已暂停的 Trigger 组的信息 |
| es\_scheduler\_state | 存储少量的有关 Scheduler 的状态信息，和别的 Scheduler 实例(假如是用于一个集群中) |
| es\_locks | 存储程序的非观锁的信息(假如使用了悲观锁) |
| es\_job\_details | 存储每一个已配置的 Job 的详细信息 |
| es\_job\_listeners | 存储有关已配置的 JobListener 的信息 |
| es\_simple\_triggers | 存储简单的 Trigger，包括重复次数，间隔，以及已触的次数 |
| es\_blog\_triggers | trigger作为 Blob 类型存储(用于 Quartz 用户用 JDBC 创建他们自己定制的 Trigger 类型，JobStore 并不知道如何存储实例的时候) |
| es\_trigger\_listeners | 存储已配置的 TriggerListener 的信息 |
| es\_triggers | 存储已配置的 Trigger 的信息 |
| es\_u\_user | 存储用户信息 |
| es\_u\_user\_role | 存储用户和角色对应关系 |
| es\_u\_role | 存储角色信息 |
| es\_u\_resc\_role | 存储访问权限配置和角色对应关系 |
| es\_u\_user\_group | 存储用户和组之间的对应关系 |
| es\_u\_resc | 存储访问权限配置 |
| es\_config | 存储一些系统相关配置项 |
| es\_code | 存储系统相关编码配置项 |
| es\_datatracking\_log | 存储数据监控日志数据 |
| es\_logs | 存储用户和系统的操作日志 |
| es\_repeat\_alarm | 存储重复告警设置 |
| es\_async\_job\_queue | 存储异步任务队列 |
| es\_report\_job | 存储天维度任务报表 |
| es\_report\_job\_rt | 存储详细维度任务报表 |
| es\_job\_user\_sub | 存储用户订阅任务 |

# FAQ

为什么集群时为实例 ID 使用 AUTO

AUTO 为专门为集群准备的。不幸的是，在某些早期的版本，1.4.5 版所用的机制任何情况下都不会清理旧的实例 ID。1.5.1 及以后版中有一些插入式的实例 ID 生成器，其中一个是基于节点的 IP 地址来生成 ID；只要你在一台给定的机器上仅有一个 Quartz 集群节点时这个生成器工作的很好。集群时应当使用 AUTO，因为很多人把 Quartz 放在 EAR 中，分布部署在集群的应用服务器上。这时候，EAR 中必然只有一个 quartz.properties 文件，因此它在所有的节点上是相同的。假如实例 ID 是硬编码的，Quartz 集群将不能正常工作，因为所有的节点有着一样的 ID。AUTO 就是为解决这个问题的。  
一些其他严重的集群问题在 Quartz 1.5.1 被引入。如果你需要对 Quartz 集群，需要至少是1.5.1以后的版本。  
EasySchedule V1.0中使用Quartz1.80版本。

在不同的机器上运行节点

EasySchedule不关心你是在相同的还是不同的机器上运行节点。当集群是放置在不同的机器上时，通常称之为水平集群。节点是跑在同一台机器是，称之为垂直集群。对于垂直集群，存在着单点故障的问题。这对高可用性的应用来说是个坏消息，因为一旦机器崩溃了，所有的节点也就被有效的终止了。

在集群环境中的全局监听器

集群环境中EasySchedule使用了quartz的全局监听器。他遵守以下回调规则：Job 或 Trigger 是在哪个 Scheduler 实例上执行的，通知的就是这个 Scheduler 实例上的监听器。因为 Job 和 Trigger 只会在单个节点上执行，也就只会通知那个节点上的监听器。

Job和StatefulJob区别

StatefulJob的JobDataMap 在每次执行之后重新持久化到 JobStore 中  
两个或多个有状态的 JobDetail 实例不能并发执行