**Tomcat及服务器状态监控系统**

**概要设计说明**

项目组长：郑丁公

**小组成员：张弢，张俊杰，寿嘉能**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | **文件标识：** | Tomcat及服务器状态监控系统-《概要设计说明（修改）》 |
| **当前版本：** | 1.1 |
| **作 者：** | 张弢 |
| **完成日期：** | 2018.7.18 |

目录

[1 引言 3](#_Toc28440)

[1.1 编写目的 3](#_Toc29051)

[1.2 范围 3](#_Toc22351)

[1.3 定义 3](#_Toc9487)

[1.4 参考资料 5](#_Toc11144)

[2 总体设计 5](#_Toc16543)

[2.1 需求规定 5](#_Toc24774)

[2.2 运行环境 6](#_Toc5951)

[2.2.1.最低系统配置 6](#_Toc20347)

[2.2.2推荐系统配置 6](#_Toc29310)

[2.3 基本设计概念和处理流程 6](#_Toc19007)

[2.3.1部署图 6](#_Toc21717)

[2.3.2流程图 7](#_Toc17139)

[2.5人工处理过程 8](#_Toc17574)

[2.6 尚未解决的问题 8](#_Toc14939)

[3 接口设计 9](#_Toc3071)

[3.1 短信接口 9](#_Toc21817)

[4 运行设计 9](#_Toc31686)

[4.1 运行模块组合 9](#_Toc2913)

[4.2 运行控制 9](#_Toc29945)

[4.3 运行时间 9](#_Toc18537)

[5 系统数据结构设计 9](#_Toc20767)

[5.1 逻辑结构设计要点 9](#_Toc21216)

[5.1.1 数据库设计 9](#_Toc19236)

[5.1.2实现模式 12](#_Toc13872)

[5.2 物理结构设计要点 12](#_Toc21557)

[6 系统出错处理设计 13](#_Toc3222)

[6.1 出错信息 13](#_Toc23639)

[6.2 补救措施 13](#_Toc12757)

[6.3 系统维护设计 14](#_Toc5322)

[7 界面原型设计 14](#_Toc27793)

# 1 引言

## 1.1 编写目的

系统设计是开发人员进行的工作，他们将系统设计阶段得到的目标系统的逻辑模型转换为目标系统的物理模型，该阶段得到工作成果――系统设计说明书是下一个阶段系统实施的工作依据。预期的读者是软件开发人员，由于这只是课程设计的一部分，因此是在小组里是所有的小组人员。

## 1.2 范围

**项目名称**：Tomcat及服务器状态监控

**任务提出者**：翁文勇老师

**用户：Tomcat使用者**

**开发者**：张弢，张俊杰，郑丁公，寿嘉能

项目提出的背景：目前，Tomcat虽然有监控系统，但是对于系统的状态仅仅是监控，所以为了可以在监控过程中发现超出正常状态的情况时可以让用户知道这个信息，本项目可以监控并且发出警告信息给用户达到一个新的监控效果。

**该软件系统同其他系统或其他机构的基本的相互来往关系:**该系统主要运行在WINDOWS 98以上的WINDOWS系列操作系统上。 因此需要有WINDOWS操作系统的支持，底层的所有工作包括声音依靠WINDOWS的声音处理API，图像处理全部依靠WINDOWS的GDI。该系统属于应用软件系统，因此处于应用层，在有这些操作系统层的支持下，不需要其它额外的支持。由操作系统负责与该系统用到的输入输出打交道。

**目标**：文案第一章部分必须全部完成，之后部分待定

## 1.3 定义

**Spring**

Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson 在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。 简单来说，Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

**SpringMVC**

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring MVC 分离了[控制器](http://baike.baidu.com/view/122229.htm" \t "https://blog.csdn.net/w15053392535/article/details/_blank)、模型[对象](http://baike.baidu.com/view/2387.htm" \t "https://blog.csdn.net/w15053392535/article/details/_blank)、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

**MyBatis**

 MyBatis 本是[apache](http://baike.baidu.com/view/28283.htm" \t "https://blog.csdn.net/w15053392535/article/details/_blank)的一个开源项目[iBatis](http://baike.baidu.com/view/628102.htm" \t "https://blog.csdn.net/w15053392535/article/details/_blank), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。MyBatis是一个基于Java的[持久层](http://baike.baidu.com/view/198047.htm" \t "https://blog.csdn.net/w15053392535/article/details/_blank)框架。iBATIS提供的[持久层](http://baike.baidu.com/view/198047.htm" \t "https://blog.csdn.net/w15053392535/article/details/_blank)框架包括SQL Maps和Data Access Objects（DAO）MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Old Java Objects，普通的 Java对象）映射成数据库中的记录。

**Maven**

Maven的核心功能便是合理叙述项目间的依赖关系，通俗点讲，就是通过pom.xml文件的配置获取jar包，而不用手动去添加jar包

**依赖传递**

依赖传递的意思就是，A项目 依赖 B项目，B项目 依赖 C项目，当使用A项目时，就会把B也给加载进来，这是传递依赖，依次类推，C也会因此给加载进来。依赖传递有好处，也有坏处，坏处就是jar包的冲突问题

**Scope**

依赖范围，意思就是通过pom.xml加载进来的jar包，来什么范围内使用生效，范围包括编译时，运行时，测试时

**SSM 框架集**

SSM（Spring+SpringMVC+MyBatis）框架集由Spring、SpringMVC、MyBatis三个开源框架整合而成，常作为数据源较简单的web项目的框架。

**MySQL**

MySQL是一个[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "https://baike.baidu.com/item/mySQL/_blank)，由瑞典MySQL AB 公司开发，目前属于 [Oracle](https://baike.baidu.com/item/Oracle" \t "https://baike.baidu.com/item/mySQL/_blank) 旗下产品。MySQL 是最流行的[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "https://baike.baidu.com/item/mySQL/_blank)之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "https://baike.baidu.com/item/mySQL/_blank)的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是[开放源码](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E7%A0%81/7176422" \t "https://baike.baidu.com/item/mySQL/_blank)这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

**JAR**

JAR（Java Archive，Java 归档文件）是与平台无关的文件格式，它允许将许多文件组合成一个[压缩文件](https://baike.baidu.com/item/%E5%8E%8B%E7%BC%A9%E6%96%87%E4%BB%B6/7121310" \t "https://baike.baidu.com/item/JAR/_blank)。为 J2EE 应用程序创建的 JAR 文件是 EAR 文件（企业 JAR 文件）。

## 1.4 参考资料

CSDN博客

百度百科

Github

# 2 总体设计

## 2.1 需求规定

**精度**

计算数据为双精度，其它的为整型或字符型。

**时间特性要求**

a． 响应时间:30ms

b． 更新处理时间:50ms；

c． 数据的转换和传送时间:10ms内；

d. 数据刷新时间：1min；

**输入输出要求**

输入与输出：用户通过输入用户名，密码，端口号进入Tomcat及服务器状态监控系统。系统给出所有关于此服务器的状态图。同时通过阿里云短信服务发送出可能发出的警告信息。

## 2.2 运行环境

### 2.2.1.最低系统配置

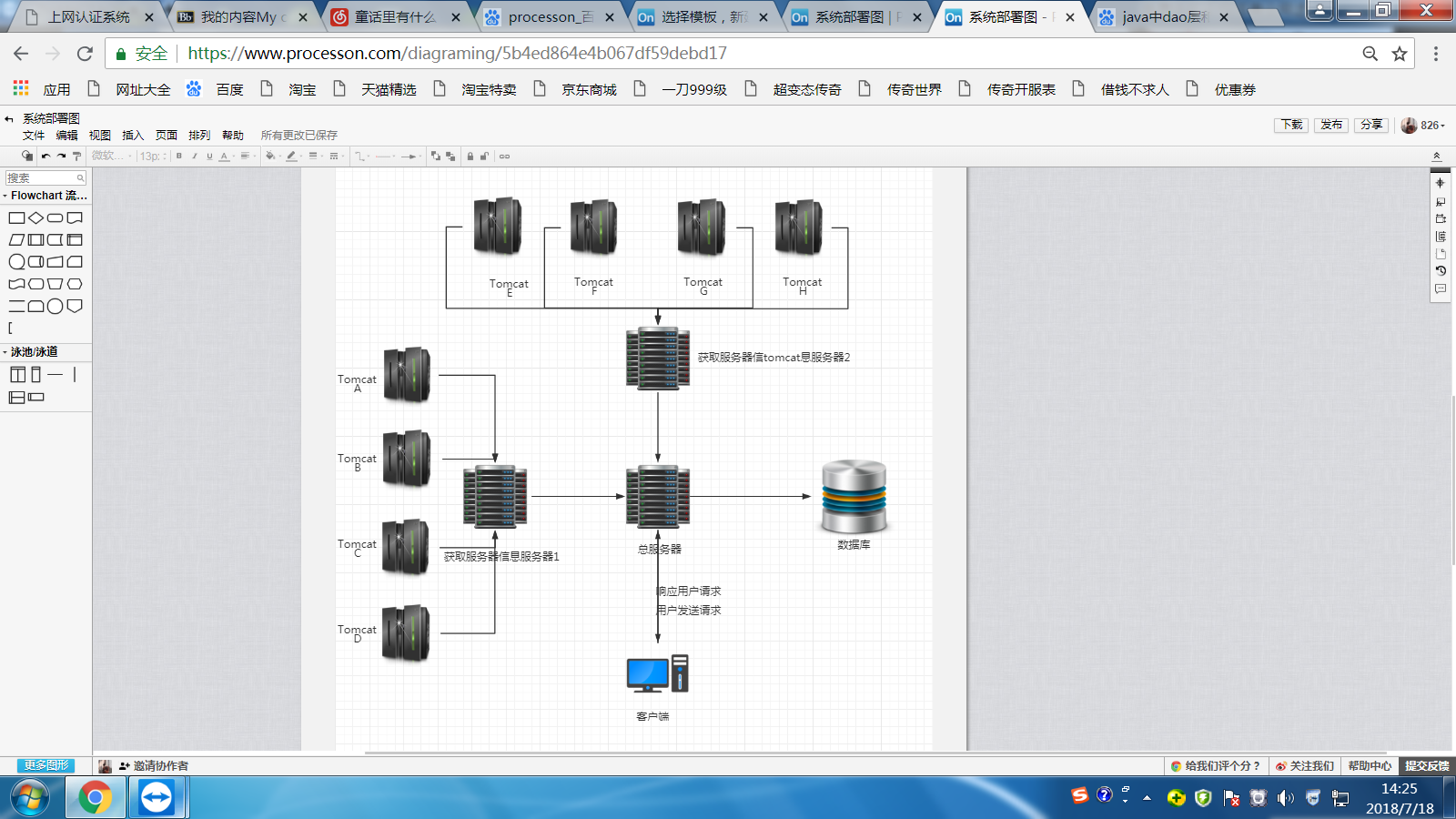
|  |  |
| --- | --- |
| **系统** | Windows 98 / 98SE / Me / 2000 / XP 中文版 |
| **CPU** | Intel PentiumIII 800 MHz 以上 |
| **内存** | 128 MB 以上 |
| **显卡** | 分辨率 1024×768 以上高彩 |
| **声卡** | 兼容 DirectSound 声卡 |
| **硬盘** | 可用空间 100 MB 以上 |

### 2.2.2推荐系统配置

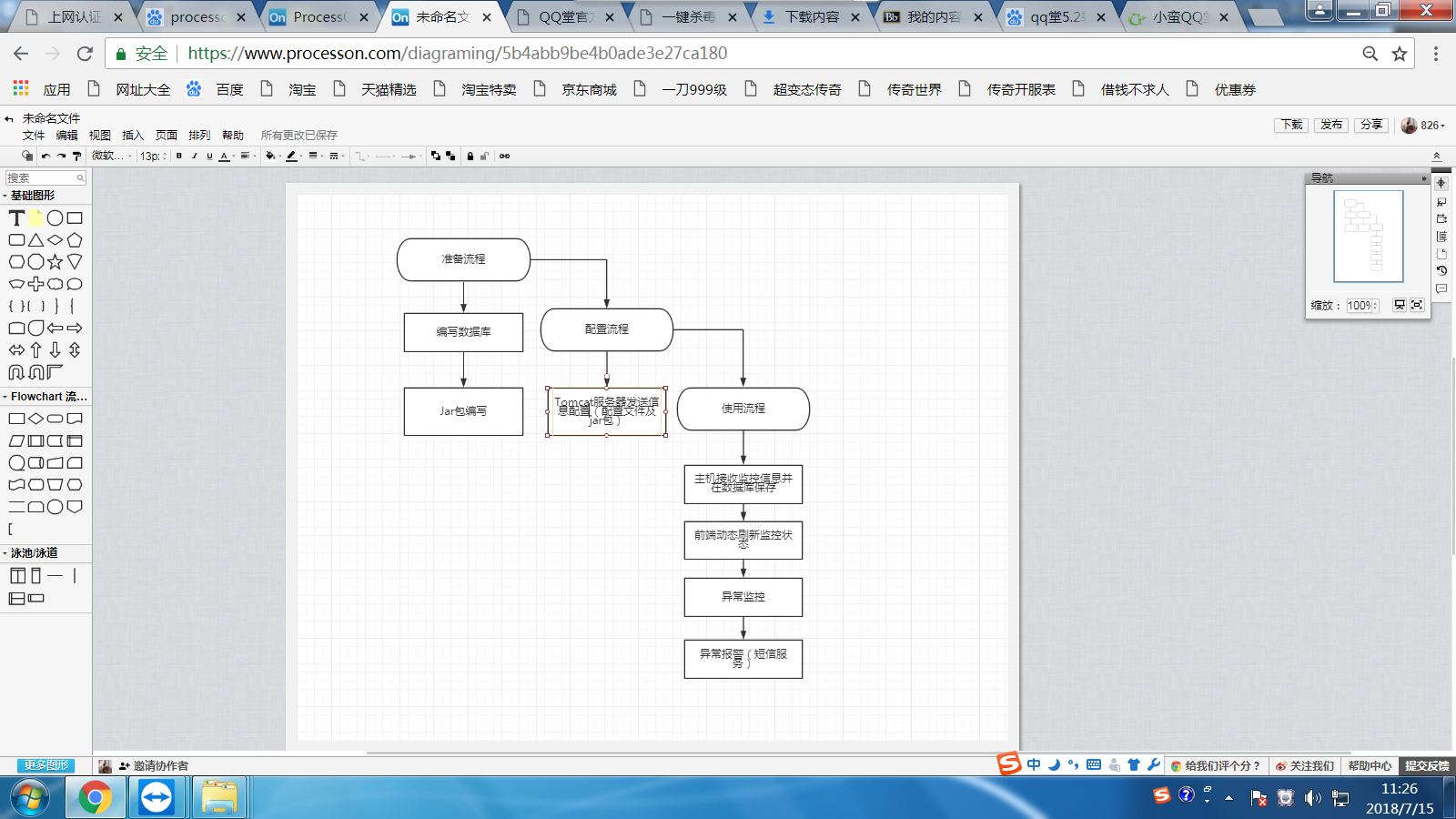
|  |  |
| --- | --- |
| **系统** | Windows XP 中文版 |
| **CPU** | Intel Pentium4 1.5 GHz 以上 |
| **内存** | 256 MB 以上 |
| **显卡** | 分辨率 1024×768 以上真彩 |
| **声卡** | 兼容 DirectSound 声卡 |
| **硬盘** | 可用空间 500 MB 以上 |

## 2.3 基本设计概念和处理流程

### 2.3.1部署图



### 2.3.2流程图



## 

## 2.5人工处理过程

【说明在本软件系统的工作过程中不得不包含的人工处理过程（如果有的话）。】

## 2.6 尚未解决的问题

【说明在概要设计过程中尚未解决而设计者认为在系统完成之前必须解决的各个问题。】

1）了解整个软件的需求分析。

2）对于软件的数据结构和整体关系进行全面了解。

# 3 接口设计

## 3.1 短信接口

短信接口用来将日志表的信息发送给阿里云服务器，由阿里云服务器进行短信报警功能。

# 4 运行设计

## 4.1 运行模块组合

【说明对系统施加不同的外界运行控制时所引起的各种不同的运行模块组合，说明每种运行所历经的内部模块和支持软件。】

## 4.2 运行控制

【说明每一种外界的运行控制的方式方法和操作步骤。】

配置文件：进入被监控的Tomcat安装文件夹，将配置文件添加并且进行自行修改。

登录监控系统：待定。

监控IP切换：在网页左侧IP栏进行选择。

## 4.3 运行时间

【说明每种运行模块组合将占用各种资源的时间。】

# 5 系统数据结构设计

## 5.1 逻辑结构设计要点

### 5.1.1 数据库设计

1. **配置表config**

|  |  |  |  |
| --- | --- | --- | --- |
| *名称* | *注释* | *数据类型* | *主要的* |
| ipAddress | IP地址 | VARCHAR(20) | TRUE |
| port | 端口 | int(11) | TRUE |
| memory\_total | 内存总量阈值百分比 | double | FALSE |
| memory\_pool\_Compressed\_Class\_Space\_used | 内存池Compressed\_Class\_Space已使用阈值百分比 | double | FALSE |
| connector\_currentThreadsCount | 线程池当前线程阈值（最大线程总量）百分比 | double | FALSE |
| connector\_currentThreadsBusy | 线程池当前繁忙线程（最大线程总量）阈值百分比 | double | FALSE |
| memory\_pool\_Code\_Cache\_used | 内存池Code\_Cache已使用阈值百分比 | double | FALSE |
| memory\_pool\_Metaspace\_used | 内存池Metaspace已使用阈值百分比 | double | FALSE |
| memory\_pool\_PS\_  Eden\_Space\_used | 内存池PS\_Eden\_Space已使用阈值百分比 | double | FALSE |
| memory\_pool\_PS\_  Old\_Gen\_used | 内存池PS\_Old\_Gen已使用阈值百分比 | double | FALSE |
| memory\_pool\_PS\_  Survivor\_Space\_  used | 内存池PS\_Survivor\_Space已使用阈值百分比 | double | FALSE |

1. **连接表connecting**

|  |  |  |  |
| --- | --- | --- | --- |
| *名称* | *注释* | *数据类型* | *主要的* |
| ipAddress | IP地址 | VARCHAR(20) | TRUE |
| port | 端口 | int(11) | TRUE |
| status | 状态 | VARCHAR(20) | FALSE |
| name | 名称 | VARCHAR(50) | FALSE |
| last\_time | 最后一次发信息的时间 | timestamp | FALSE |

1. **memory\_pool**

|  |  |  |  |
| --- | --- | --- | --- |
| *名称* | *注释* | *数据类型* | *主要的* |
| Name | 角色ID | VARCHAR(50) | TRUE |
| TYPE | 角色名称 | VARCHAR(50) | FALSE |
| Initial | 初始化使用内存 | double(20,0) | FALSE |
| Committed | 当前分配内存总量 | double(20,0) | FALSE |
| Maximum | 当前最大能分配的内存总量 | double(20,0) | FALSE |
| Used | 当前已使用内存 | double(20,0) | FALSE |
| Port | 端口 | int(11) | TRUE |
| ipAddress | IP地址 | VARCHAR(20) | TRUE |
| Time | 时间 | timestamp | TRUE |

1. **connector**

|  |  |  |  |
| --- | --- | --- | --- |
| *名称* | *注释* | *数据类型* | *主要的* |
| Name | 角色ID | VARCHAR(50) | TRUE |
| maxThreads | 最大线程数 | int(11) | FALSE |
| currentThreadCount | 当前线程数 | int(11) | FALSE |
| currentThreadsBusy | 当前繁忙线程数 | int(11) | FALSE |
| maxTime | 最大处理时间 | int(11) | FALSE |
| processingTime | 进程处理时间 | bigint(11) | FALSE |
| requestCount | 需要的线程数 | int(11) | FALSE |
| errorCount | 出错的线程数 | int(11) | FALSE |
| bytesReceived | 收到的字节 | bigint(11) | FALSE |
| bytesSent | 发送的字节 | bigint(11) | FALSE |
| time | 时间 | timestamp | TRUE |
| ipAddress | IP地址 | VARCHAR(20) | TRUE |
| port | 端口 | int(11) | TRUE |

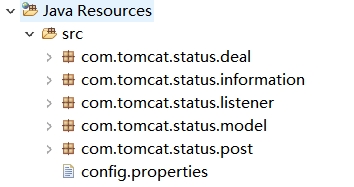
1. **日志表log**

|  |  |  |  |
| --- | --- | --- | --- |
| *名称* | *注释* | *数据类型* | *主要的* |
| ipAddress | IP地址 | VARCHAR(20) | TRUE |
| port | 端口 | int(11) | TRUE |
| time | 时间 | timestamp | TRUE |
| error\_type | 错误类型 | VARCHAR(50) | TRUE |
| describe\_message | 描述信息 | VARCHAR(200) | FALSE |
| read\_status | 读取状态 | VARCHAR(20) | FALSE |
| send\_status | 发送状态 | VARCHAR(20) | FALSE |

1. **memory**

|  |  |  |  |
| --- | --- | --- | --- |
| *名称* | *注释* | *数据类型* | *主要的* |
| ipAddress | IP地址 | VARCHAR(20) | TRUE |
| Free | 当前空余内存量 | double | FALSE |
| Total | 当前已分配内存量 | double | FALSE |
| Max | 最大分配内存量 | double | FALSE |
| Port | 端口 | int(11) | TRUE |
| Time | 时间 | timestamp | TRUE |

### 5.1.2实现模式



|  |  |
| --- | --- |
| Java层 | 作用 |
| Deal | 通过调用information获取Tomcat的信息，包装成model中的类，通过post方法传输到服务器后台 |
| Information | 获取Tomcat服务器状态信息 |
| Listener | 监听器 |
| Model | 数据类型 |
| Post | 传输方法 |
| 注释：整体采用json格式 | |

## 5.2 物理结构设计要点

【给出本系统内所使用的每个数据结构中的每个数据项的存储要求，访问方法、存取单位、存取的物理关系（索引、设备、存储区域）、设计考虑和保密条件。】

# 6 系统出错处理设计

## 6.1 出错信息

【用一览表的方式说明每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。】

|  |  |  |  |
| --- | --- | --- | --- |
| 软件故障 | 数据异常 | 异常无报警短信 | 页面卡死 |
| 原因 | 1. 可能服务器未能将数据传入 2. 可能服务器传入数据与数据库预设不符 3. 可能数据库数据调用出错 4. 可能数据库超出最大储存量 | 1. 异常不在预期异常内 2. 阿里云服务器错误 3. 无需警报的异常 | 1. 开启线程过多 2. 刷新速度过快 3. 数据信息过多 |
| 故障处理要求 | 1. 检查配置文件及jar包 2. 检查数据库字段 3. 检查调用文件 4. 检查数据库内存 | 1. 通知管理员 2. 联系阿里云 | 1.增加监控系统信息处理服务器数量 |
| 后果 | 无反应 | 阿里云短信服务不运行 | 存档读档文件损坏 |
| 硬件故障 | 无显示 | 无显示 | 显示错误或异常 |

## 6.2 补救措施

1. 若在安装期间出现错误导致程序无法执行，可以检查机器的性能进行再次安装。

2. 在硬件方面要保证机器能够稳定的运行。

3. 当出现数据丢失或日志丢失。到备份的数据库或日志信息中获取部分存档信息。

## 6.3 系统维护设计

【说明为了系统维护的方便而在程序内部设计中做出的安排，包括在程序中专门安排用于系统的检查与维护的检测点和专用模块。】

系统的维护和管理也是一个系统的问题，也包括两个层面：

1）技术层面 包括：（1）使用较新的开发和维护技术； （2）专职维护人员维护硬件

2）人员层面 包括：

（1）维护协议； （2）思想认识； （3）法律法规教育；

（4）工作流程及方式方法。

# 7 界面原型设计