**环境空气质量在线监测平台**

**概要设计说明书**



编制：

校对：

审核：

审定：

批准：

日期： 年 月 日

**西安思坦科技有限公司**

**环境空气质量在线监测平台**

**概要设计说明书**



**西安思坦科技有限公司**

目录

[1 引言 1](#_Toc510099811)

[1.1 编写目的 1](#_Toc510099812)

[1.2 项目背景 1](#_Toc510099813)

[1.3 参考资料 1](#_Toc510099814)

[2 整体架构 2](#_Toc510099815)

[2.1 系统描述 2](#_Toc510099816)

[2.2 设计原则 2](#_Toc510099817)

[2.3 物理架构 4](#_Toc510099818)

[2.3.1 物理拓扑 4](#_Toc510099819)

[2.3.2 运行环境 5](#_Toc510099820)

[2.3.2.1 软件环境 5](#_Toc510099821)

[2.3.2.2 硬件环境 6](#_Toc510099822)

[2.3.3 开发环境 7](#_Toc510099823)

[2.3.3.1 软件环境 7](#_Toc510099824)

[2.3.3.2 硬件环境 7](#_Toc510099825)

[2.4 逻辑架构 7](#_Toc510099826)

[2.5 数据交互流路 8](#_Toc510099827)

[2.5.1 数据上行流路 8](#_Toc510099828)

[2.5.2 数据下行流路 9](#_Toc510099829)

[3 子系统设计 10](#_Toc510099830)

[3.1 远程交互系统设计 10](#_Toc510099831)

[3.1.1 领域整体分析设计 11](#_Toc510099832)

[3.1.1.1 领域整体分析 11](#_Toc510099833)

[3.1.1.2 服务整体设计 12](#_Toc510099834)

[3.1.2 连接器架构设计 13](#_Toc510099835)

[3.1.3 通信使能机制设计 13](#_Toc510099836)

[3.1.4 现场端代理架构设计 15](#_Toc510099837)

[3.1.4.1 逻辑层连接建立流程 16](#_Toc510099838)

[3.1.4.2 反控流程 18](#_Toc510099839)

[3.1.4.3 数据接收流程 19](#_Toc510099840)

[3.1.4.4 现场端与现场端代理交互设计 19](#_Toc510099841)

[3.1.4.5 功能接口设计 20](#_Toc510099842)

[3.1.4.6 应用协议模块设计 21](#_Toc510099843)

[3.1.4.7 污染物在线监控（监测）系统数据传输标准 21](#_Toc510099844)

[3.2 数据入库与存储系统 22](#_Toc510099845)

[3.2.1 入库系统设计 22](#_Toc510099846)

[3.2.2 存储系统设计 24](#_Toc510099847)

[3.2.2.1 实体图 25](#_Toc510099848)

[3.2.2.2 E-R图 28](#_Toc510099849)

[3.3 在线监控应用系统 29](#_Toc510099850)

[3.3.1 逻辑架构 29](#_Toc510099851)

[3.3.2 功能结构 31](#_Toc510099852)

[3.3.2.1 系统管理 31](#_Toc510099853)

[3.3.2.2 站点管理 32](#_Toc510099854)

[3.3.2.3 实时数据 32](#_Toc510099855)

[3.3.2.4 历史数据 33](#_Toc510099856)

[3.3.2.5 报表分析 33](#_Toc510099857)

[3.3.2.6 数据导出 33](#_Toc510099858)

[3.3.2.7 GIS定位 33](#_Toc510099859)

[3.3.2.8 AQI 34](#_Toc510099860)

[3.3.2.9 预警预报 34](#_Toc510099861)

[3.3.2.10 溯源分析 34](#_Toc510099862)

[3.3.3 数据架构 34](#_Toc510099863)

[3.3.3.1 数据结构 34](#_Toc510099864)

[3.3.3.2 数据操作 34](#_Toc510099865)

[3.3.4 界面设计 37](#_Toc510099866)

[3.3.4.1 界面内容设计 38](#_Toc510099867)

[3.3.4.2 导航设计 39](#_Toc510099868)

[3.3.4.3 界面实现 40](#_Toc510099869)

[3.3.4.4 界面展示 40](#_Toc510099870)

[4 安全设计 41](#_Toc510099871)

[4.1 安全认证技术 41](#_Toc510099872)

[4.2 数据加密处理 42](#_Toc510099873)

[4.3 权限控制 42](#_Toc510099874)

[4.4 日志和安全审计 43](#_Toc510099875)

[4.5 事务处理设计 43](#_Toc510099876)

[4.6 数据备份 44](#_Toc510099877)

[4.7 网页防篡改设计 44](#_Toc510099878)

[4.8 防火墙技术 44](#_Toc510099879)

[5 出错异常设计 45](#_Toc510099880)

[5.1 异常分类 45](#_Toc510099881)

[5.2 异常处理 45](#_Toc510099882)

[6 测试方案 47](#_Toc510099883)

[6.1 测试要求 47](#_Toc510099884)

[6.2 测试环境 47](#_Toc510099885)

[6.3 测试数据 47](#_Toc510099886)

[6.4 测试策略 48](#_Toc510099887)

[6.4.1 功能测试 48](#_Toc510099888)

[6.4.2 用户界面(UI)测试 48](#_Toc510099889)

[6.4.3 性能测试 49](#_Toc510099890)

[6.4.4 安全性测试 49](#_Toc510099891)

[6.4.5 兼容性测试 50](#_Toc510099892)

[7 实施管理 50](#_Toc510099893)

[7.1 开发管理 50](#_Toc510099894)

[7.2 版本管理 51](#_Toc510099895)

[8 评审意见 53](#_Toc510099896)

# 引言

## 编写目的

本文档根据空气质量监测系统的需求说明书，定义了系统的主要功能模块及子系统相互之间的联系，并定义了模块的技术实现方法。

本文档定义软件总体设计结构，确定软件子系统，I/O接口，处理模式，数据库实现设计，界面设计，安全设计等。从各个角度用符号化的方法保证项目下一步更好进行 。

本文档的预期读者为：项目经理、设计人员、开发人员、测试人员。

## 项目背景

随着社会发展节奏的加快，环境问题日益凸现，对空气质量监测的要求也逐渐增加。空气质量在线监测能快速地了解某地区、某城市的环境空气质量现状，可以使环境空气质量状况透明化，促进环境空气质量的改善。我国的环境空气质量监测是于上世纪70年代逐步开展起来的，但是直到2000年，我国才结束了环境空气质量自动监测系统组成仪器全部依赖进口的被动局面。因此，我国自主开发的适合我国国情的环境空气质量自动监测系统目前还有较大的发展空间。

根据国务院办公厅关于生态环境监测网络建设的要求，依据此需求，即需要开发一套可实现高密度网格化布局的低成本、多参数集成的紧凑型微型环境空气监测系统，网格化的监测体系可在区域内全覆盖，实现高时空分辨率的大气污染监测，结合信息化大数据的应用实现污染来源追踪、预警预报等功能，为环境污染防控提供更为及时有效的决策支持。

## 参考资料

《空气质量监测系统平台需求分析》

《环境保护应用软件开发管理技术规范》（HJ622-2011）

《污染源在线自动监控监测系统数据传输标准212-2005》

《环境污染源自动监控信息传输、交换技术规范》（HJ-T352-2007）

《计算机软件文档编制规范》GBT 8567-2006

《环境信息系统集成技术规范》HJT 418-2007

《环境信息系安全技术规范》HJ 729-2014

# 整体架构

## 系统描述

空气质量在线监测平台，采用多层次的系统设计方法，可以对接不同性质（国控，省空，区域等），不同厂家的空气质量子站相关数据，建立一套完善的空气质量监测、预警、发布的可视化平台。同时用数据质控，远程反控、统计分析等信息化手段，帮助环境监测部门及时、全面、准确地掌握本辖区的空气质量现状，实现对本辖区监测站点空气质量进行准确分析，为空气质量的溯源提供决策平台。建立统一的公众发布信息平台，提供以互联网，手机APP，微信等多种多样的发布形式和发布渠道。

## 设计原则

（1）兼容性

系统的建设中既要考虑用户的当前的业务需求，又要考虑和已有的应用系统的有效的衔接、集成，同时还要考虑如何能和日后可能要建设的系统集成和整合，因此，系统的设计要提高系统的兼容性，可以和其他应用系统高度的整合在一起达到信息共享。

（2）稳定性

系统建设采用先进和高度商品化的软硬件平台、网络设备和开发工具。在进行系统设计、实现和测试时采用科学有效的技术和手段，确保系统交付使用后能持续稳定地运行。

（3）安全性

通过各种安全技术手段，保障系统运行的安全。遵守现行的各项保密制度和规定，对尚未公开或不宜公开的数据与信息采取严格的安全保护措施。系统具有一定的容错能力，在用户误操作或输入非法数据时不会发生错误。如在编辑等操作功能中，对于用户输入的错误信息系统能自动识别，并进行自动修复或提示用户重新输入。

系统外部安全：系统的安全性充分考虑网络的高级别、多层次的安全防护措施，包括备份系统、防火墙和权限设置等措施，保证政府部门的数据安全和政府机密；同时考虑系统出现故障时的软硬件恢复等急救措施，以保障网络安全性和处理机安全性。系统要形成相对独立的安全机制，有效防止系统外部的非法访问。

系统内部安全：在保证系统外部安全的同时，系统也能确保授权用户的合法使用。系统本身具有容错功能，包括出错提示、原因，并能自动或通过人工操作，使出错的系统恢复到正常状态。系统还提供严格的操作控制和存取控制。

系统运行安全：在逻辑上，系统具有抵御对系统的非法入侵的能力；在物理上，系统应保证不存在可能的单点故障，提供资源数据的备份能力。系统支持定期的自动数据备份和手工进行数据备份，能够在数据毁坏、丢失等情况下将备份数据倒回，实现一定的数据恢复。

（4）可维护性

维护方式：系统提供对系统自身的集中操作维护的功能，真正做到使系统能在数据损坏、丢失等情况下将备份数据倒回，实现数据恢复。

维护工作量：系统提供集中的、智能化的维护工具，尽可能减少手工维护工作量，确保系统的正常运行。

（5）易操作性

界面设计：系统提供美观实用、友好直观的中文图形化用户管理界面，充分考虑工作人员的习惯，方便易学、易于操作，含全菜单式处理和各种快捷键操作，保证多数功能一键到达。系统应以图形化的方式提供各种操作手段，充分发挥GIS以图形面对用户的特点，信息的表现方式更直观，效率更高，摆脱过去那种面对大量枯燥的表格、文字信息进行数据挖掘的状况。

（6）可扩展性

功能扩展：为了满足用户今后系统扩容和扩大应用范围的需求，系统充分考虑从系统结构、功能设计、管理对象等各方面的功能扩展。

软硬件升级：系统充分考虑软硬件平台的可扩展性及软、硬件的负载平衡机制。随着关键软件和硬件的发展以及管理功能的增加，系统具有灵活和平滑的扩展能力。

（7）开放性

为了确保应用系统具有良好的互操作性和可移植性，系统的数据格式符合有关国家标准或行业标准。

（8）先进性

系统的设计中，在保证系统稳定的前提下，选用最新的开发软件。所选用的开发工具都是当今世界上具有领导地位的成熟产品，同时具有一定的互操作性，便于系统集成，将国际上软件领域最新的成果移植于本系统，在整体设计思想上，也具有一定超前性，一方面最大限度地保护用户的现有投资，另一方面，使系统具有较强的生命力。

## 物理架构

### 物理拓扑

平台拓扑结构如图2.1所示，空气站质量监测系统平台通过环保专网接收在线监测仪器传输的监测因子数据并存储到数据库中。

监控中心网络设备主要包括防火墙，路由器、核心交换机以及应用服务器，数据库服务器，监控中心平台使用的防火墙用于平台服务器的安全，路由器和核心交换机方便以后的网络的扩展，服务器作为平台数据的采集、存储以及Web访问的硬件平台。另外，UPS为整个平台硬件提供稳定的电源。应用服务器用来部署软件系统，WEB服务器部署前端web应用，数据库服务器存放平台数据；这两个服务器同时可采用云部署方式，对应的环境参考2.4和2.5。

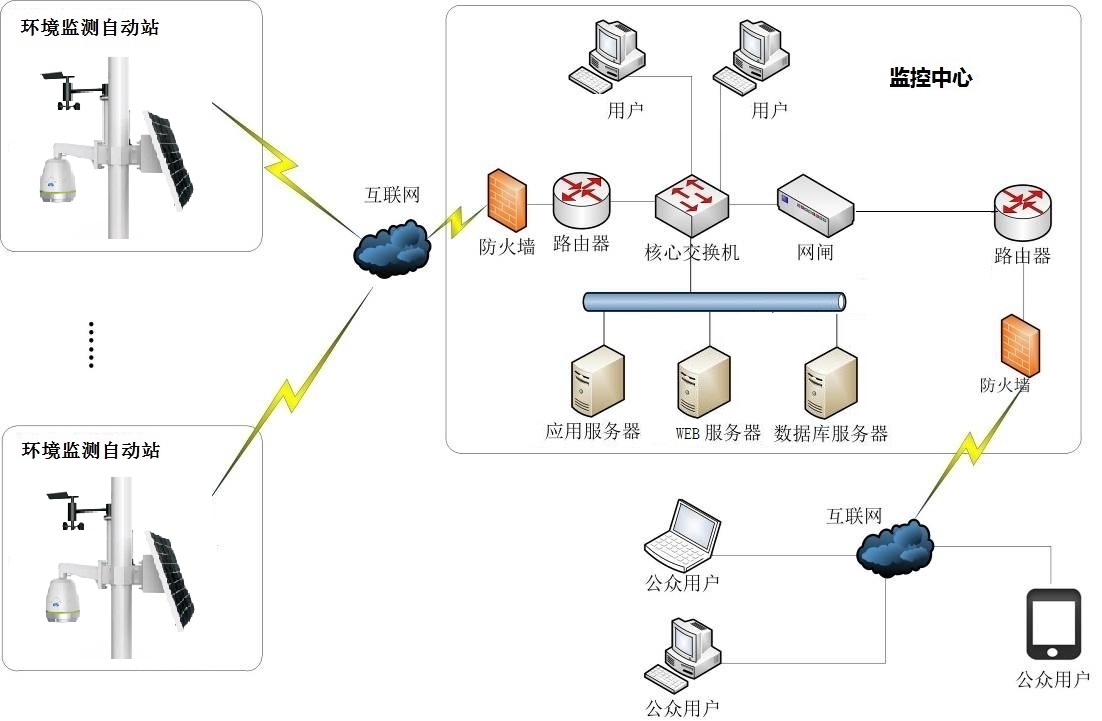


图2.1 平台物理架构

### 运行环境

系统开发完成后，需要部署到服务器上运行来支持客户端的访问，软硬件环境如下。

#### 软件环境

|  |  |  |
| --- | --- | --- |
| **分类** | **名称** | **语种** |
| 操作系统 | Linux/Windows | 简体中文 |
| 数据库 | Mysql | 简体中文 |
| 服务器 | tomcat | 简体中文 |

#### 硬件环境

空气站质量在线监测系统数据平台的的硬件环境有传统硬件部署和云平台部署两种可选方案。

* **传统部署方式**：
* **应用服务器**：部署空气站质量在线监测系统数据平台软件。
* **Web服务器**： 部署web网站，向浏览器等Web客户端提供文档和服务， 也可以放置网站文件和数据文件，让用户浏览。
* **数据库服务器**：部署数据库软件，保证各个检控点的数据量交互，同时还能够确保其他方面的数据访问要求。
* **企业网络**：企业需具备区域网络，支持子公司访问，如企业使用互联网访问系统，则需具备固定IP及域名。
* **云平台部署方式**
* **云服务器：**部署空气站质量在线监测系统数据平台软件和相关数据。

云平台部署的优点：

1. 数据上传到云端，[提供](http://www.aliyun.com/zixun/aggregation/1148.html)了最可靠、最[安全](http://www.aliyun.com/zixun/aggregation/47.html)的[数据存储](http://www.aliyun.com/zixun/aggregation/235.html)中心。用户不用再担心数据丢失、病毒入侵等麻烦。
2. 对用户端的设备要求最低。用户不需要高性能的电脑来运行基于Web的应用。
3. 更强大的计算能力。 当连接到一个云系统，你就拥有了可自行支配的整个云的力量。用户不再局限于单台计算机所能做的事情，利用成千上万台计算机和服务器的能力，来执行超级计算类的任务。

无限的存储容量，可根据不同阶段的需求购买不同的存储容量，避免了资源浪费。

### 开发环境

#### 软件环境

|  |  |  |
| --- | --- | --- |
| **分类** | **名称** | **语种** |
| 操作系统 | Windows 7 | 简体中文 |
| 数据库 | Mysql | 简体中文 |
| 服务器 | tomcat | 简体中文 |
| 开发工具 | MyEclipse | 简体中文 |
| 开发语言 | Java | 英文 |

#### 硬件环境

一台装有开发环境的Windows 电脑。

## 逻辑架构

平台系统涉及的参与者包括操作人员、现场端设备以及其他系统，所需提供功能较多，业务逻辑较为复杂，同时可能会随环境因素的变化而需要改变，因此需要具备灵活性，可扩展性与可伸缩性。

根据平台系统涉及的参与者，可将功能划分为与操作人员相关，与现场端相关以及数据存储三大类。同时，由于平台的业务逻辑的复杂性，将平台根据功能分解为各个子系统，将平台解耦合可以更加突出平台的灵活性和可扩展性，设计依据如下：

1）根据核心功能性需求相关性将系统划分为远程交互系统、数据存储系统、应用系统三个子系统。远程交互系统主要实现与现场端的数据交互；数据存储系统实现对接收到的有效数据进行存储与访问；应用系统提供业务相关的功能的实现与接口；

2）根据灵活性的质量属性需求，采用数据库作为数据系统平台内部数据传输的唯一方式。



图2.2 软件逻辑架构

## 数据交互流路

### 数据上行流路

Web应用系统在浏览器显示数据的流程：



图2.3 数据上行流图

### 数据下行流路

数据下行流路主要指的是平台的反控功能，数据流向图如下：



图2.4 数据下行流图

# 子系统设计

## 远程交互系统设计

远程交互系统中，现场端与服务中心端（以下简称中心端）建立连接，中心端接收现场端数据进行存储，对web应用端提供服务接口，web端调用该接口进行反控，中心端与现场端交互，完成反控功能。



服务器要完成的工作：处理连接及并发、处理与每个现场端的交互过程、数据处理（解析入库等）、向web端发布功能服务接口。

### 领域整体分析设计

#### 领域整体分析

服务器为现场端提供服务，并为web端提供接口，也是数据的处理中心将远程交互子系统中的服务器端也成为中心端。现场端与中心端的通信主要分为三类：

1. 现场端主动发起连接请求与中心端建立起底层通信连接；



1. 在应用层现场端主动向中心端发送数据/报警信息；



1. 在应用层中心端主动向现场端发送反控指令。



在现场端与中心端建立起底层通信连接后，中心端（现场端）作为应用层交互的主动方或者被动方受到外部影响，具有随机性。

#### 服务整体设计

建立连接和数据交互实质上是相互独立的，建立连接只负责与远程端主机建立连接关系，并不关心后续具体的交互处理过程。

在业务逻辑较为简单的情况下，比如只要求服务器接收和打印日志的日志服务器，将二者统一处理即可。而本系统的业务逻辑较为复杂，交互过程与数据处理可能耗时很长，因此，将底层通信连接的建立与应用层的交互分离，使用代理模式对应用层的交互逻辑进行封装。



服务器将任务分解，只需要处理连接及并发问题，具体与每个现场端的交互任务与数据处理交由现场端代理完成。

将服务器提供的功能进行分解设计，系统结构图如下。



连接器负责建立连接，然后交由现场端代理处理后续交互流程。

### 连接器架构设计

连接器实质上是一个仅对客户端（现场端）建立连接请求进行接收与处理的网络服务器。连接器的核心任务是建立连接并对连接进行处理，但对后续处理的具体步骤不明确。

服务器可以分为循环式，并发式，和反应式等几大类。在这一设计空间中，需要权衡的主要在于：是要简化编程，还是要提高程序的可伸缩性。连接器需要具备处理多个现场端的并发连接，因此循环式不满足并发与实时性要求；并发式服务器在内部定义并执行新线程，灵活性不足；反应式服务器满足实时性、并发性、灵活性的特点。

因此，采用反应式服务器架构设计，抽象连接处理接口。





### 通信使能机制设计

在连接器与现场端建立起底层通信连接后，需求将建立的通信链路调用接口传递给现场端代理，使现场端代理具备与现场端通信的能力。同时应用层的通信应尽量降低与具体通信机制之间的耦合。

考虑到以下几点：

1）现场端与服务器之间的通信底层采用面向连接的TCP协议，因此可以通过socket接口实现对现场端数据的收发。只需要将服务器被动创建的可用socket对象传递给现场端代理就可以实现现场端代理与现场端之间的交互；

2）Socket属于较低层的系统通信机制，现场端代理主要功能是处理逻辑事务，将其与具体的通信机制绑定不利于灵活性实现；

因此，定义抽象接口（communicationBase）对底层通信链路接口封装后传递给现场端代理。接口定义抽象的数据读写与异常处理等相关接口。从而实现底层通信实现与现场端代理之间的灵活性。



不同应用场景下连接数量（并发量）可能存在较大差异，连接服务器的具体实现可以采用系统层级通信接口实现，也可以使用第三方通信框架，因此设计抽象的通信接口对具体通信实现进行封装。

底层TCP/IP的通信主要包括读写操作。将读写功能封装成接口，具体应用协议根据该接口封装成对应的调用接口，其他模块只需调用接口即可完成通信功能。



### 现场端代理架构设计

现场端代理是对现场端硬件设备及其功能的抽象，每个连接到服务器的现场端设备都会在服务器端生成一个与之对应的代理对象，作为与现场端交互的接口。同时代理对象对每个业务功能的实现逻辑，数据通信等细节进行了封装。

对于现场端代理的设计需要考虑以下几点：

1）代理需要向上层应用系统提供反控功能调用接口；

2）代理需要对具体的应用层协议进行封装；

3）代理可能会随着现场端设备的变化而修改功能与实现。

4）代理需要在逻辑上现场端建立映射关系。

因此，采用分层的设计模式将代理模块划分为接口层和逻辑实现层。





#### 逻辑层连接建立流程

连接服务器实现了网络通信层连接的建立，可以通过业务通信对象与现场端进行通信，但在应用协议层无法将业务通信对象作为对现场端设备的标识，因此需要一定的方式将服务器接收创建的业务通信对象与对应的现场端标识进行映射绑定。

中心端与现场端建立连接需要考虑以下几点：

1）中心端与现场端之间存在一对多的关系；

2）现场端可以通过通信层连接识别中心端，中心端无法通过通信层连接识别现场端；

3）应用层协议不包含对通信层标识（socket）的识别，只有应用层的识别标识（MN编码）；

4）需要建立应用层标识与通信层标识之间的映射。

在通信层连接建立后现场端主动发送包含应用层识别码的数据，中心端收到数据后将应用层识别码与通信层标识绑定。



#### 反控流程



#### 数据接收流程



#### 现场端与现场端代理交互设计

通信交互过程是指具体协议中规定的现场机与现场端代理之间的数据收发过程。对于不同的协议，具体任务的交互过程也随之改变，因此，通信交互方案应具备一定的稳定性，实现协议可变更，具有通用性。

由于现场机与现场端代理连接过程中，数据发送的随机性，不能确定某时刻接收的数据是否为本次应接收的数据，因此，必须解决每一次交互任务中，发送数据后，如何接收正确的响应帧。

将命令交互的过程分成异步处理，发送请求后，异步处理应答等接收到的数据，进行统一处理。将协议进行分类，收到数据后判断类别，然后按类别处理。

Processor完成具体的处理任务，包括数据解析入库等处理。每个任务命令对应一个processor处理过程，提供具体的发送请求接口与处理接口。并使用运行状态字来标识处理的阶段，并完成各个阶段的具体的处理过程（解析，入库等）。



* **方案优点**

（1）通用通信处理框架，异步处理的过程使得协议可自定义（目前使用的212协议也可以兼容2005版本与2017版本）；

（2）所有接收数据进行统一处理，对于异常信息只是作为其中一类进行处理，使异常情况的处理变得更简单。

#### 功能接口设计

代理需要向应用系统提供功能调用接口，而代理所提供的功能由具体的现场端硬件功能所确定。提供的功能可能会随着硬件设备功能的变动而产生变化，而应用系统应该保持一定的稳定性，因此，代理功能调用接口需要具备灵活性与可扩展性，以保证在代理功能发生变化时应用系统所做出的变动尽可能小。



使用一个统一的动态接口封装所有功能接口，将功能的名称与相关数据作为接口函数的参数，从而确定要调用的具体功能。

这样，即使功能产生变化，该模块结构也无需改变，对外发布的动态接口不变，只需要修改内部功能接口，而对于客户端，只是调用时传入的参数改变了，外部调用方的改动量也较小。

使用面向服务（SOA）的思想，将所提供的功能接口封装成服务，客户端根据提供的服务接口直接调用相关服务实现相关功能。

#### 应用协议模块设计

现场代理需要使用应用层通信协议与现场端进行交互，应用层协议存在变化的可能性。应用层协议改变时对现场端代理产生的影响应尽可能小。

使用规范化调用接口来尽量缩小不同协议之间的差异性。



对外提供规范化的统一接口，调用接口可以实现协议帧的解析封装，外部不必关心协议的具体实现。若更改协议，只需更改协议实现，接口不变。

#### 污染物在线监控（监测）系统数据传输标准（212）

标准适用于污染物在线监控（监测）系统、污染物排放过程（工况）自动监控系统与监控中心之间的数据传输，规定了传输的过程及参数命令、交互命令、数据命令和控制命令的格式，给出了代码定义。



## 数据入库与存储系统

数据入库与存储系统将远程交互系统提供的数据进行处理入库，架构如下所示。



### 入库系统设计

* **问题描述**

入库系统需要入库的数据由两部分组成，一部分是远程交互系统传输的现场端数据，一部分是在线监测应用系统用户的输入数据。

* **问题分析**

入库系统处于两个应用中间，会出现两个应用同时操作数据的情况，所以必须保障数据的一致性。

入库系统处于两个应用的中间，起到信息交互的作用，考虑到两个应用都与入库系统有直接的关系，将入库系统独立出来并提供方便可用的接口是最佳的实现方法。

* **解决方案**

使用目前成熟的数据层接口实现方法，将数据抽象为DAO持久层，使用dao设计模式封装数据的增删改查等操作来实现入库系统。

mybatis持久层框架来实现数据的操作，不但可以将数据操作封装成接口，还可以使用自带的事务管理功能保证数据的一致性。

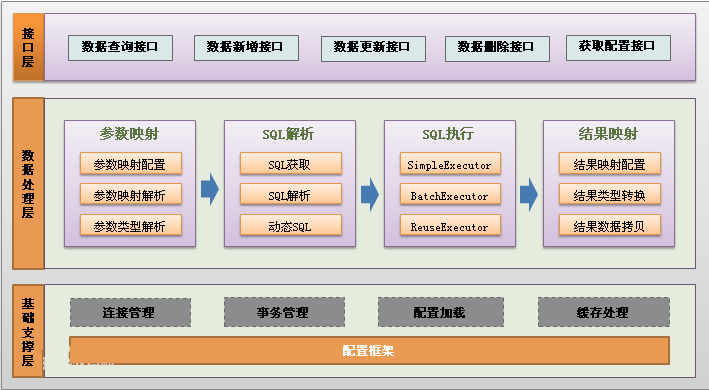
* **技术介绍**

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，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。

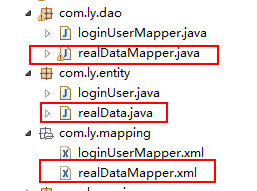
MyBatis优点：

* 简单易学：没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件；易于学习，易于使用。
* 灵活：mybatis不会对应用程序或者数据库的现有设计强加任何影响。 sql写在xml里，便于统一管理和优化。
* 解除sql与程序代码的耦合：将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。
* 提供映射标签，支持对象与数据库字段的关系映射；

MyBatis的整体框架如下图所示：



* **接口举例**



### 存储系统设计

* **问题描述**

（1）入库系统从远程交互系统和在线监控应用系统获取到的数据是无组织的数据，没有相应的存储方式和结构。

（2）存储的数据不仅包括存储从下位机接受的实时数据，还需要保存web客户端用户的相关信息。

* **问题分析**

（1）无组织的数据比较混乱，平台各个功能模块需要从众多无组织的数据中筛选后，方可进行二次使用，费时费力。

（2）如何将实时数据与历史数据分隔开进行管理并消除数据库冗余。

* **解决方法**

（1）通过对用户需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型，并采用自底向上的方法．用 E-R 图表示各实体之间的联系。

（2）在关系型数据库中建立实时表和历史表，实时表用于更新操作，历史表用于插入操作，保持实时表的大小不变。

#### 实体图

通过需求分析我们得到设备等最基本的实体：设备信息，监测数据，站点信息等，画出它们的实体图如下：

1. 用户信息实体图，如下图所示：



用户信息实体图

1. 用户权限实体图，如下图显示：



用户权限实体图

1. 站点信息实体图，如下图所示：



站点信息实体图

1. 数据信息实体图，如下图所示：

数据信息实体图

数据信息是一个综合性的概念，包括实时数据，分钟数据，小时平均值日报表数据，日平均值月报表数据以及月平均值日报表数据，其中实时数据是基础数据，其他各个报表的数据都需要通过数据库触发器自动生成所需数据信息，故各个不同的报表数据信息结构相同。

1. 报警信息实体图，如下图所示：



报警信息实体图

1. 日志实体图，如下图所示：



图 运行日志实体图

1. 维护日志实体图，如下图所示：



维护日志实体图

#### E-R图

实体与实体间有多种联系，画出各种实体间的联系图如下：

1. 用户信息表与维护日志表的联系图，如下图所示：

用户与维护日志ER图

1. 站点信息与日志的联系图，如下图所示：

站点与日志ER图

1. 站点信息与报警信息的联系图，如下图所示：

站点与报警信息ER图

1. 站点信息与数据信息的联系图，如下图所示：



图 站点信息与数据信息ER图

1. 实时数据、日报表、月报表以及年报表的数据信息的联系图，如下图所示：

实时数据、日报表、月报表以及年报表ER图

## 在线监控应用系统

在线监控应用系统设计为B/S系统，需要实现从前端到后台的所有设计，为了扩展性要求，需要预留下可供调用的接口。由于开发者经验不足，使用成熟的架构实现系统的设计是最好的解决方法。

### 逻辑架构

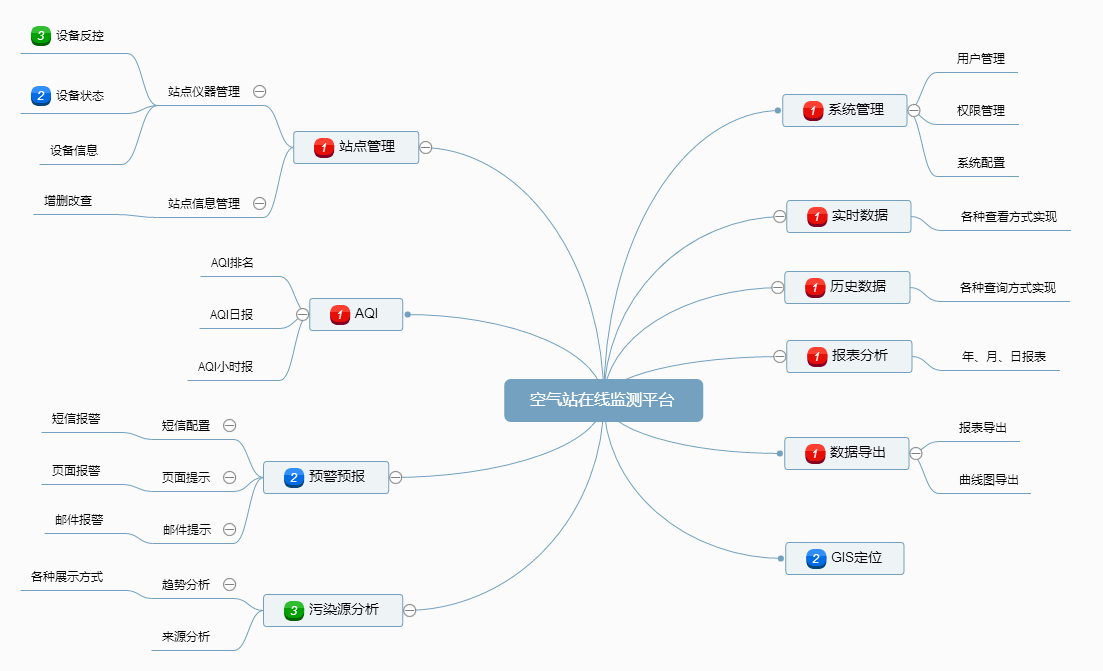
在线监控应用系统由前端显示层，交互接口层，业务逻辑层，系统资源层组成，严格按照MVC的设计模式来实现。

 依据MVC的架构设计，设计在线监控应用系统如下图所示。其中展示层也就是显示层展示给用户看到的界面，对应MVC中的“V”，交互接口层是为了实现面向服务的设计预留的两个类型的接口，业务逻辑层主要实现MVC架构的模型层，将各个功能模块设计成对应的实体对象和工具类，对应MVC中的“M”；业务接口层完成业务逻辑层和数据库之间的功能实现，也就对应了“C”控制层。系统资源层为应用平台的各层之间提供数据支撑。

### 功能结构

空气质量在线监测平台是可以在任何一台有网的电脑上输入对应的IP和项目名访问的B/S系统，通过该平台数据中心可提供所属地区各监测点位数据的实时采集传输、实时监控空气环境质量，实现在线数据查询及报表统计、数据自动预警、环保信息综合分析、数据归集和排名反馈等，为环保的研究提供信息资源和手段，为环保业务管理提供统一的管理平台。

考虑到日后其他同类型项目的需求，系统需要同时具备可扩展性，易于维护的特性，将各个功能进行模块化划分，可以降低系统的复杂度和开发困难，并增强可维护性，整个系统的模块设计采用低耦合，高内聚的设计依据。



#### 系统管理

（1）用户管理

对于不同的角色设置相应权限管理，一个角色关联了一套操作权限。系统共提供了三种操作权限。

系统用户：拥有系统的所有功能操作权限；

管理用户：拥有部分业务相关的功能操作权限；

普通用户：只能进行系统中相关内容的查询操作，实现不同级别操作人员对数据访问范围和数据读写性的严格控制，建立统一用户管理平台实现所有用户的身份管理，包括用户个人身份信息、角色信息、电子邮箱、个人账号和密码。

（2）系统配置

配置与系统相关的一些信息，如短信配置。

此功能可以查看短信配置详情，添加条目可以新增加短信推送人员信息和发送内容，编辑选项可对接收短信用户推送内容进行管理操作，配置的信息内容包括预警信息、日报、状态预警、掉线预警，完成设置以后，列表中人员可以收到短信信息。

（3）日志管理

用户的操作日志，系统运维日志等信息。

#### 站点管理

（1）站点信息管理

用户在此模块可以实现监测点位信息的增、改、查、删等基本操作，点位信息包括监测点位名称、地址、经纬度、站点ID、所在区域名称等内容，实现点位信息的动态管理，区域与编号为锁定状态，可自行配置名称、经纬度、排名、公开、掉线预警等选项。

（2）站点设备管理

系统可以实现实时监视在线监测仪器是否正常工作，数据上传是否正常，从而清楚设备的运行状况及运行进度，当前端数据采集设备或仪器出现故障时，系统自动提供报警信息方便站点负责人及时知晓，并采取相应的解决措施，保证系统的正常、稳定运行。

#### 实时数据

1. 站点数据实时查看

用户点击监测点位自动显示空气质量指数AQI和各项监测因子实时数据，空气质量指数AQI数值与表示颜色搭配显示，直观表现站点当前污染情况，同时可以按照不同需求配置需要显示的监测因子，显示时间段分为实时状态值、最近一小时值、最近24小时值等。

1. 远程视频实时监控

监测现场可以安装视频监控设备，通过窗口视图直观了解监测站点的周边情况和污染物实时排放数据，以保证系统运行的稳定性。当数据异常提醒之后，可以通过回传影像资料判断现场情况（需人工进行），当发生不可抗力因素时，同样可以根据影像资料来判定事故详情。

#### 历史数据

历史数据是以实时数据为依托的，用户点击历史数据，查询监测点位的空气质量指数AQI和各项监测因子历史数据，同时可以按照不同需求配置需要显示的监测因子，显示时间段根据用户想要查询的时间确定。

#### 报表分析

系统根据历史数据和一定的算法生成对应的日月年报表。根据报表分析出该监测点位的空气质量趋势，给执法机构提供监测支撑。

#### 数据导出

系统提供空气质量、气象数据导出功能，用户在设置时间类型、站点、时间段以后即可实现数据导出，内容包括点位信息、数据更新时间、常规6参数浓度值、主要污染物、空气质量指数AQI。其中数据有效率按照国家标准进行计算，分钟值以后端数据传输判定为准，小时值以每小时收集45个分钟值为准，日均值以每天收集22个小时值为准，其余时间区间以日均值有效天数为准。数据导出设计有多种导出形式,，如pdf,word,image等。

#### GIS定位

系统内所有监测点位按所属行政区域进行展示，方便用户直观、一目了然掌握各个行政区域监测点位的部署情况，同时监测点位颜色按空气质量指数AQI表示颜色动态显示，系统提供多种方式的地图效果（矢量、卫星、三维）来实时显示空气子站的位置和实时数据。

#### AQI

按照HJ633-2012环境空气质量指数(AQI)技术规定要求，生成实时报、日报，月报等数据报表，发布的指标包括各监测站点的监测站点信息、空气质量指数（AQI）、首要污染物、空气质量指数类别以及空气质量指数说明等信息，可生成word、Excel、PDF多种格式的报表。

#### 预警预报

系统提供预警、日报通知功能，预警包括超标预警、断线预警和异常值预警，在监测数值超标、数据连接中断和出现异常值时，自动给设定联系人发送提醒信息，保证系统的正常、稳定运行，日报通知将辖区内各个行政区空气质量指数日均值以短信形式发送给站点负责人或主管领导，让环境管理者及时掌握环境空气质量变化情况，在空气质量恶化时第一时间知道详细信息。

#### 溯源分析

收集点位数据后，平台对各项污染物统计值进行计算分析，初步建立点位污染源模型（拟采用方法为首要污染物比重饼状图解析），如果监测点位条件允许，能够实现现场采样，则可以更加精确的进行污染物对比分析，通过各时间段污染物比重模型结合地区现状来分析具体污染源和现场实际情况，并提供针对性治理方案。

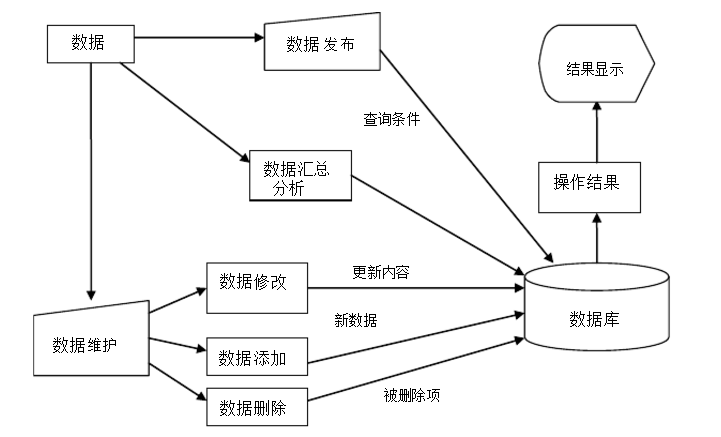
### 数据架构

#### 数据结构

在线监测应用系统的数据存储，同步等见4.2；

#### 数据操作

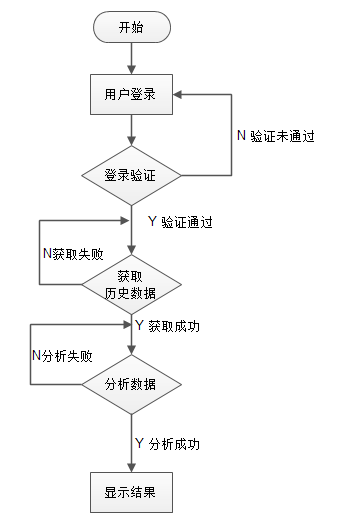
数据操作如下图所示：



数据管理：



数据分析：



### 界面设计

空气质量监测系统的界面设计遵循用户控制式，以用户为主导。用户在界面启动行为，发送给程序，程序做出反应并给用户以反馈。

界面设计主要体现表现层的任务，就是用来与用户进行交互，接收用户的操作请求并按照要求返回所请求的结果。环境空气质量在线监测管理系统的表现层设计采用网站常用的厂字型结构，顶部为网站功能菜单，下方部分的左侧为导航栏功能菜单，右侧为详细的信息展示界面，对应表现层的每一个功能页面。前台文件使用DIV+ CSS设计实现样式，JS 脚本响应基本操作和表单数据的校验，列表数据的显示利用 GridView 控件或表格，地图与图表显示以Echarts为主，前台与后台的数据交互以及特效刷新展示则由 AJAX 技术实现。



界面设计图

#### 界面内容设计

内容设计主要是将Web应用系统的可视内容展现在用户的Web浏览器上。Web应用系统的内容设计必须与空气站质量监测的性质和宗旨相匹配。只有当访问者从Web服务器请求网页后，环境空气质量在线监测平台的网页内容才最后确定。

1. 表单设计

表单是用户与Web服务器之间请求联系的枢纽，在网页界面设计时，对表单进行如下考虑设计原则：

* 决定域类型：是否允许使用任意文本，是否只限于预定义的值，是否为必填值。
* 确定域的适当大小：考虑界面的美观度。
* 制定域的应用格式：文本居左，数字居右。
* 如果应用程序逻辑允许，保留浏览器上对域值得复制，粘贴的功能。
* 对域的对齐位置进行标记。
* 尽可能使用下拉列表，如果列表项太多，使用带滚动条的弹出列表代替下拉列表。
* 尽量使用复选框和单选框替代输入框，减少用户输入错误。
* 用视觉上有吸引力的方式对域进行分组。

1. 图表设计

图表是数据展示的重要方式，数据产品最离不开的其实就是图表设计，既要保证图表本身数据清晰准确、直观易懂，又要在找准用户关注的核心内容进行适当的突显，帮助用户通过数据进行决策。精准表达图表中的数据需要做到如下：

* 选择合适的图表

首先要明确每种图表的定义和使用范围，其次要使用正确的方式将图表绘制出来，不能在表达上有所偏差。明确每种图表的定义和使用场景有助于对数据进行加工和二次传达。根据柱形图、饼图、折线图的特点，选择合适的图表类型。

* 正确的绘制图表

图表强调数据的准确性，错误的坐标轴选择，或者关键元素的缺失，会导致图表的准确性下降，表意不明。

#### 导航设计

导航设计主要设计用户的动作，也涉及应用程序的代码，Web应用系统的导航以活动链接的形式存在。应用程序网页之间的导航要精心策划，无论是直观的还是页面上的导航面板，导航必须有明确的用户可以理解的逻辑，使用户不会在网页的“超空间”中迷失。环境空气质量在线监测平台主要是以查看和检索操作为目的的应用，需要严格设计好导航栏。

菜单和链接是网页之间导航的2个主要工具，都可以实现由当前页面进入到另一个页面的功能。

面包屑提供当前页所在位置的可视化，并用于导航到此页，通常放在页面内容的上方，导航项随用户在页面之间的导航而改变，并且允许用户后退到以前访问的页面。

#### 界面实现

环境空气质量在线监控平台通过界面开发工具和相应的技术实现前端界面的展示，便于用户获取所需的信息。

1. 界面开发工具

利用Dreamweare制作出跨越平台限制和跨越浏览器限制的充满动感的网页。 使用网站地图可以快速制作网站雏形，设计，更新和重组网页。改变网页位置或档案名称，Dreamweare 会自动更新所有链接。使用支援文字，HTML码，HTML属性标签和一般语法的搜寻及置换功能使得复杂的网站更新变得迅速又简单。

在网站前端开发过程中需要把图片用Photoshop处理成加载到网站页面的图片用Photoshop 制作矢量图形用作网站页面的修饰按钮。

1. 界面技术实现

环境空气质量监测平台主要实现数据的可视化，采用Echarts插件来实现地图与图表的显示，DIV+CSS实现页面布局，JavaScript网页增加动态功能，AJAX实现和后台数据交互，使用JQuery进行插件开发。

#### 界面展示



登录界面



界面结构布局



图表展示

# 安全设计

## 安全认证技术

我们在基于 Intranet / Internet 的一些应用中，通过 SSL（Secure Sockets Layer [安全套接层](https://baike.baidu.com/item/%E5%AE%89%E5%85%A8%E5%A5%97%E6%8E%A5%E5%B1%82)）安全套接口技术，保证通信双方都是可信任的及通信过程中信息的不可篡改性，保证双方身份的不可抵赖性。

## 数据加密处理

我们在设计中将对关键敏感信息（如用户口令等）进行加密处理，尤其是在外网上的应用，其关键数据将被加密之后再送入数据库中，保证数据库层面没有关键敏感信息的明码保存，保证在数据库存储层的安全性。同时在网上通信传输过程中，敏感信息将使用 SSL安全套接口技术，保证网络传输层的安全性。

## 权限控制

我们设计中将所有应用逻辑都集成在中间应用服务器层，通过严格的权限控制进行数据存取。权限控制的另一个方面是应用系统的授权使用，我们将保证用户所需要的服务，均在用户身份认证库进行校验，并根据执行权限进行控制。

使用shiro进行权限控制的实现。Apache Shiro是一个强大且易用的Java安全框架，它 可以非常容易的开发出足够好的应用，其不仅可以用在 JavaSE 环境，也可以用在JavaEE 环境。Shiro 可以帮助我们完成：认证、授权、加密、会话管理、与 Web 集成、缓存等功能，使用Shiro的易于理解的API,您可以快速、轻松地获得任何应用程序,从最小的移动应用程序到最大的网络和企业应用程序。

Shiro具有以下特点：

* 易用性 ：提供易于理解的 Java Security API。
* 广泛性 ：可以为你的安全需求提供“一站式”服务。
* 灵活性 ：Apache Shiro可以工作在任何应用环境中。Shiro既不强加任何规范，也无需过多依赖。
* Web能力 ： Apache Shiro允许基于应用URL和Web协议（如REST）创建灵活的安全策略，同时还提供了一套控制页面输出的JSP标签库。
* 可插拔 ：Shiro干净的API和设计模式使它可以方便地与许多的其他框架和应用进行集成。

## 日志和安全审计

所有用户访问记录将记载在中心服务器，供系统管理员备查。在系统中提供安全审计工作，安全审计主要记录用户操作行为的过程，用来识别和防止网络攻击行为、追查网络泄密行为并用于电子举证的重要手段。对用户的越权访问进行预警。

日志采用了log4j技术，Log4j是一个用Java编写的可靠，快速和灵活的日志框架（API），它在Apache软件许可下发布。

Log4j是高度可配置的，并可通过在运行时的外部文件配置。它根据记录的优先级别，并提供机制，以指示记录信息到许多的目的地，诸如：数据库，文件，控制台，UNIX系统日志等。

Log4j中有三个主要组成部分：

* **loggers:** 负责捕获记录信息。
* **appenders :** 负责发布日志信息，以不同的首选目的地。
* **layouts:** 负责格式化不同风格的日志信息。

## 事务处理设计

事务(Transaction)是访问并可能更新数据库中各种数据项的一个程序执行单元(unit)。事务将若干的数据库操作作为一个整体控制,一起成功或一起失败。

事务具有以下特性：

原子性：指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

一致性：指事务前后数据的完整性必须保持一致。

隔离性：指多个用户并发访问数据库时，一个用户的事务不能被其他用户的事务所干扰，多个并发事务之间数据要相互隔离。

持久性：指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，即时数据库发生故障也不应该对其有任何影响。

我们将充分应用数据库系统和Spring框架提供的事务处理技术，保证数据库中数据的完整性、一致性。

## 数据备份

一个安全的数据备份机制是保证系统所有数据的安全性的重要手段，鉴于此系统拥有大量的数据，建议采取“统一备份”的机制，做到及时、安全的数据备份，同时减轻了数据备份的工作量。

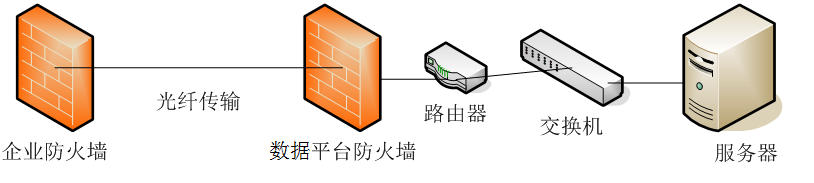
## 网页防篡改设计

网页防篡改系统是完全保护 Web 网站不发送被篡改内容并进行自动恢复的Web页面保护软件。网页防篡改系统采用先进的 Web 服务器核心内嵌技术，将篡改检测模块（数字水印技术）和应用防护模块（防注入攻击）内嵌于 Web 服务器内部，并辅助以增强型事件触发检测技术，不仅实现了对静态网页和脚本的实时检测和恢复，更可以保护数据库中的动态内容免受来自于 Web的攻击和篡改，彻底解决网页防篡改问题。篡改检测是使用密码技术，为网页对象计算出唯一性的数字水印。公众每次访问网页时，都将网页内容与数字水印进行对比；一旦发现网页被非法修改，即进行自动恢复，保证非法网页内容不被公众浏览。同时，应用防护也对用户输入的 URL 地址和提交的表单内容进行检查，任何对数据库的注入式攻击都能够被实时阻断。

## 防火墙技术

网络检测站主要包括企业原有的防火墙，平台使用的防火墙，路由器、核心交换机以及服务器，企业原有的防火墙主要负责原有整个厂区内(即集团办公大楼、各分公司及职工宿舍)的网络安全，平台使用的防火墙用于平台服务器的安全，路由器和核心交换机方便以后的网络的扩展，服务器作为平台数据的采集、存储以及Web访问的硬件平台。另外，UPS为整个平台硬件提供稳定的电源。机房原有大型UPS为调度中心和厂区的视频服务器不间断供电。

信息中心平台网络拓扑图如图4.1所示。



主控中心网络拓扑图

# 出错异常设计

空气质量在线监测系统是企业级应用，对应着复杂的业务逻辑，当系统出现或可能出现异常情况时，需对异常情况进行分析、处理。系统中一个很小的漏洞，也会造成整个软件开发系统的垃圾资源无限增多，最终可能导致系统瘫痪，因此必须对系统中的资源进行合理管理，捕获各种可能出现的异常状况，确保系统稳定运行。

## 异常分类

根据系统应用,常见错误异常分为数据存储部分、应用部分、核心库部分、商业层部分。

数据存储部分:主要是系统在与数据库交互时,常见的一些错误.比如数据库连接错误,数据库对象不存在或数据字符过多。

 应用部分:主要是反映用户在键盘输入操作时,可能引起的数据类型错误,字符长度超过限制等,使用鼠标或键盘可能引起的操作错误等。

 核心库部分:主要反映系统框架中的一些错误。比如数组下标越界,数字超出范围等。

商业层部分:主要反映系统中的一些如权限被拒绝,输入参数错误等。

通信部分：网络连接异常等。

## 异常处理

异常尽量在编码时进行处理，通常java语言处理异常的方法是采用try-catch包裹可能异常的代码。

由Try代码块保护的代码所发生的任何异常，甚至包括在不含Try代码块的被调函数或方法以内的异常都将被Catch代码块内的代码处理。当然，除非catch代码块自己也扔出了异常，而在这种情况下异常会被扔到下一个级别更高的try代码块，哪怕那意味着从当前正在执行的函数中扔出异常。

异常的4条重要规则：

* 如果catch块有若干个，那么异常将根据其类型被扔给最适当的一个catch接受处理。
* 如果没有找到可接受的catch块，则异常被从当前的try块扔到从调用顺序链中找到的下一个可用的catch块。
* 异常对象的类型给出了发生错误本质的重要信息。
* 异常可以通过throw关键词显式扔出。

下表中简单列出了各类异常的处理办法

|  |  |
| --- | --- |
| 分类 | 处理 |
| 数据存储部分 | 异常中主要使用SqlException类 |
| 应用部分 | 对于用户输入错误在软件界面层中进行具体的分析，判断错误类型，具体提示信息由软件开发人员,软件设计人员及系统相关人员进行商议，最后输出错误信息到AppError.resources文件。 |
| 核心库部分 | 该部分包括系统核心模型。根据其应用不同,其消息也有所不同。最后输出错误信息到AppError.resources文件。 |
| 商业层部分 | 输出错误信息到AppError.resources文件。 |
| 通信部分 | 对于网络连接超时或异常需进行超时判断并加入重发机制，如果出现异常连接输出异常信息； |

# 测试方案

## 测试要求

软件的错误是不可避免的，所以必须经过严格的测试。通过对软件的测试， 尽可能的发现软件中的错误，借以减少系统内部各模块的逻辑，功能上的缺陷和错误， 保证每个单元能正确地实现其预期的功能。检测和排除子系统（或系统）结构或相应程序结构上的错误，使所有的系统单元配合合适，整体的性能和功能完整。并且使组装好的软件的功能与用户要求保持一致。

## 测试环境

* **硬件环境**

一台装有开发环境和测试环境的PC测试机。

* **软件环境**

|  |  |
| --- | --- |
| **资源名称**/**类型** | **配置** |
| 操作系统环境 | 操作系统主要分为 windows XP，windows 7。其中 windows XP 和windows 7 是重点测试对象 |
| 浏览器环境 | IE（IE8 以上）、 Chrome、 Firefox、360等主流浏览器。 |
| 功能性测试工具 | Junit（对程序代码进行单元测试的一种Java框架） |

## **测试数据**

本测试方案的测试数据来源于数据库中的模拟数据和后期终端传输后的数据。

## 测试策略

系统测试类型及各种测试类型所采用的方法、工具等介绍如下：

### 功能测试

|  |  |
| --- | --- |
| 测试范围 | 验证数据的精确度、数据类型、业务功能等相关方面的正确性。 |
| 测试目标 | 核实所有功能均已正常实现，即是否与需求一致。 |
| 采用技术 | 主要采用白盒测试方法。 |
| 工具与方法 | Junit |
| 开始标准 | 开发阶段对应的功能完成并且测试用例设计完成 |
| 完成标准 | 测试用例通过并且最高级缺陷全部解决 |

### 用户界面(UI)测试

|  |  |
| --- | --- |
| 测试范围 | 1.导航、链接、 Cookie、页面结构包括菜单、背景、颜色、字体、按钮名称、 TITLE、提示信息的一致性等，  2.友好性、可操作性（易用性） |
| 测试目标 | 核实各个窗口风格（包括颜色、字体、提示信息、图标、 title等）都与需求保持一致，或符合可接受标准，能够保证用户界面的友好性、易操作性，而且符合用户操作习惯。 |
| 采用技术 | 网页测试通用方法 |
| 工具与方法 | 手工测试、目测(扫描) |
| 开始标准 | 界面开发完成 |
| 完成标准 | UI 符合可接受标准，能够保证用户界面的友好性、易操作性，而且符合用户操作习惯 |
| 测试重点  与优先级 | 根据实际需求而定 |
| 需考虑的  特殊事项 | 根据实际需求而定 |

### 性能测试

|  |  |
| --- | --- |
| 测试范围 | 1.系统可支持同时在线用户数；2.访问刷新时间； |
| 测试目标 | 1. 用户、管理员的密码管理  2．应用程序级别的安全性：核实用户只能操作其所拥有权限能操作的功能。  3．系统级别的安全性：核实只有具备系统访问权限的用户才能访问系统。 |
| 采用技术 | 并发测试 |
| 工具与方法 | 手工测试 |
| 开始标准 | 功能测试完成 |
| 完成标准 | 多客户端登录系统使用正常 |
| 测试重点  与优先级 | 根据实际需求而定 |
| 需考虑的  特殊事项 | 根据实际需求而定 |

### 安全性测试

|  |  |
| --- | --- |
| 测试范围 | 1.用户、管理员的密码安全；2.权限；3.非法攻击 |
| 测试目标 | 1. 用户、管理员的密码管理； 2. 应用程序级别的安全性：核实用户只能操作其所拥有权限能操作的功能。   3．系统级别的安全性：核实只有具备系统访问权限的用户才能访问系统。 |
| 采用技术 | 代码包或者非法攻击工具 |
| 工具与方法 | 手工测试 |
| 开始标准 | 功能测试完成 |
| 完成标准 | 执行各种非法操作无安全漏洞且系统使用正常 |
| 测试重点  与优先级 | 根据实际需求而定 |
| 需考虑的  特殊事项 | 根据实际需求而定 |

### 兼容性测试

|  |  |
| --- | --- |
| 测试范围 | 1．使用不同版本的不同浏览器、分辨率、操作系统分别进行测试。  2．不同操作系统、浏览器、分辨率和各种运行软件等各种条件的组合测试。 |
| 测试目标 | 核实系统在不同的软件和硬件配置中运行稳定 |
| 采用技术 | 黑盒测试 |
| 工具与方法 | 手工测试 |
| 开始标准 | 项目组移交系统测试 |
| 完成标准 | 在各种不同版本不同类型浏览器、操作系统或者其组合下均能正常实现其功能 |
| 测试重点  与优先级 | 根据实际需求而定 |
| 需考虑的  特殊事项 | 根据实际需求而定 |

# 实施管理

## 开发管理

整个项目采用java语言开发，并且采用模块化开发，每个人的开发习惯不同，开发过程中会遇到的问题如下：

1.jar包混乱:工程中存在jar包重复，版本不一致的情况。

2.添加jar包随意:需要用到第三方jar时,随便下载就添加到项目中了,没有记录。

3. 重复构建:每个开发人员从svn/git下载完代码后，都要对项目进行配置,构建。

4. 不同环境的构建差异:总是存在在开发环境运行良好的程序,到了测试环境编译不通过,或者不同环境的配置文件不一致,需要手动的去修改。

5. 模块间的开发依赖纯手动处理:提倡模块化的开发,可是怎么做好模块化开发之间的依赖处理,没有解决方案。

为解决上述问题，项目组决定使用Maven项目管理工具。

Maven是一个项目管理和综合工具。[Maven](http://www.yiibai.com/maven)提供了开发人员构建一个完整的生命周期框架。开发团队可以自动完成项目的基础工具建设，Maven使用标准的目录结构和默认构建生命周期。MAVEN以一个配置文件和中央仓库结合的方式，给java开发者开发过程中提供了方便和一定规则。

Maven的优点：

* maven提供了中央仓库，能够帮我们自动下载构件。
* maven解决了依赖的增多，版本不一致，版本冲突，依赖臃肿等问题。
* maven对于目录结构有要求，约定优于配置，用户在项目间切换就省去了学习成本。
* 项目非常大时，可借助Maven将一个项目拆分成多个工程，最好是一个模块对应一个工程，利于分工协作。而且模块之间还是可以发送消息的。

## 版本管理

版本控制系统应该是软件开发过程中除了开发工具外最基础的工具了，不管是个人开发或是团队协作开发，都可以通过版本控制系统获得巨大的好处。没有版本控制系统的话，代码可能被别人或自己不小心覆盖或遗失、也不知道是谁因为什么原因改了这段代码、也没办法可以复原回前几天的修改。有了版本控制系统，开发人员只要将每次程式码的变更都纪录（Commit）起来，并且透过版本控制系统中进行更新。

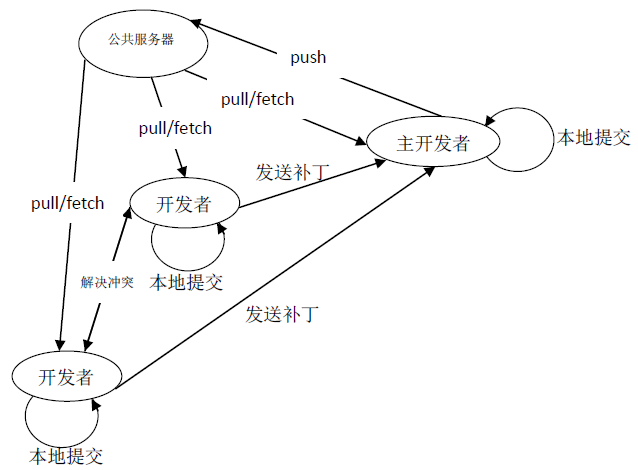
有了版本控制系统，我们可以浏览所有开发的历史纪录，掌握团队的开发进度，而且作任何修改都不再害怕，因为你可以轻易的复原回之前正常的版本。我们也可以透过分支和标签的功能来进行软件发行的不同版本，例如稳定版本、维护版本和开发中版本。

常用的版本控制有SVN和GIT。

其中SVN是集中式版本控制系统，它的特点是只有一台中央服务器，存放着所有研发数据,而其它客户端机器上保存的是中央服务器最新版本的文件快照，不包括项目文件的变更历史。所以，每个相关人员工作开始前，都需要从这台中央服务器同步最新版本,才能开始工作。但是集中式版本控制系统有很多缺点：1.网络环境要求高，相关人员必须联网才能工作。2.中央服务器的单点故障影响全局，如果服务器宕机，所有人都无法工作。3.中央服务器在没有备份的情况下，磁盘一旦被损坏，将丢失所有数据。

GIT是分布式版本控制系统，它的特点是每个客户端都是代码仓库的完整镜像，包括项目文件的变更历史。所有数据分布的存储在每个客户端，不存在中央服务器。分布式版本控制系统的优点：1.版本库本地化，版本库的完整克隆，包括标签、分支、版本记录等。2.支持离线提交，适合跨地域协同开发。3.分支切换快速高效，创建和销毁分支廉价。

根据空气站项目的需求，使用GIT更能提高工作效率，保障代码的安全性，便于负责人对代码的集中管理，git的使用如下图所示：



# 评审意见