**XXXXX系统**

**技术可行性分析报告**

项目名称：

项目编号：

编 写：

审 核：

批 准：

日 期：

目 录

[1 项目简介 ２](#_Toc298404753)

[2 系统构成 ３](#_Toc298404754)

[3 产品技术平台分析 ５](#_Toc298404755)

[3.1 系统运行环境 ５](#_Toc298404756)

[3.2 开发环境 ５](#_Toc298404757)

[4 主要关键技术 ５](#_Toc298404758)

[5 关键技术的解决方案 ７](#_Toc298404759)

[5.1 Struts2框架 ７](#_Toc298404760)

[5.2 持久层框架 １０](#_Toc298404761)

[5.3 Ajax技术 １２](#_Toc298404762)

[5.4 XML解析器 １６](#_Toc298404763)

[5.5 SNMP １９](#_Toc298404764)

[6 系统架构解决方案 ２３](#_Toc298404765)

[6.1 架构1 ２３](#_Toc298404766)

[6.2 架构2 ２３](#_Toc298404767)

[7 本项目确定的方案 ２４](#_Toc298404768)

[7.1 系统架构 ２４](#_Toc298404769)

[7.2 技术方案 ２４](#_Toc298404770)

# 项目简介

# 系统构成

|  |  |
| --- | --- |
| 模块名称 | 模块描述 |
|  |  |
|  |  |
|  |  |

# 产品技术平台分析

## 系统运行环境

|  |  |
| --- | --- |
| 网络环境： |  |
| 硬件平台： |  |
| 操作系统平台： |  |
| 数据库平台： |  |
| Web服务： |  |

## 开发环境

|  |  |
| --- | --- |
| 网络环境： |  |
| 硬件平台： |  |
| 操作系统平台： |  |
| 数据库平台： |  |
| Web服务： |  |

# 主要关键技术

|  |  |
| --- | --- |
| **主要关键技术** | **技术解释** |
| J2EE | J2EE（Java 2 Enterprise Edition）即Java 2企业版，是提供给开发者的采用组件技术构建分布式系统的编程框架。 |
| Struts2框架 | Struts2是一个兼容Struts1和WebWork的MVC框架，它是以Webwork的设计思想为核心，吸收了Struts1的优点。 |
| Spring框架 | Spring是轻量级的容器，是一个开源框架。 |
| iBatis框架 | iBatis是目前流行的轻量级持久层架构，学习上手快，使用灵活、性能高效等特点。 |
| Log4j | Log4j是Apache的一个日志记录的开放源代码项目。 |
| XML解析器 | 目前流行的XML解析器主要有DOM、SAX、JDOM、DOM4J等。 |
| WebService | Web Service也叫XML Web Service WebService是一种可以接收从Internet或者Intranet上的其它系统中传递过来的请求，轻量级的独立的通讯技术。是:通过SOAP在Web上提供的软件服务，使用WSDL文件进行说明，并通过UDDI进行注册。 |
| RMI-IIOP | 采用IIOP协议(互联网内部对象请求代理协议)进行java RMI远程方法访问。 |
| Ajax | AJAX全称为“Asynchronous JavaScript and XML”（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术。 |
| C语言 | C语言是一种高效的结构化语言。 |
| SNMP | 简单网络管理协议（Simple Network Management Protocol） |
| RRD/JRobin | RRD是Round Robin Database（环状数据库）的缩写。JRobin是一个使用Java实现的开源的RRD处理程序和绘图引擎。 |
| Flash | Flash是交互式矢量图和Web动画的标准。 |
| JNDI | Java命令与目录服务 |
| JUnit | Java单元测试的工具 |
| DOM | Document Object Model文档对象模型。 |
| CSS | Cascading Style Sheets层叠样式表单。 |
| Maven2 | Apache Jakarta 项目的高级项目管理工具，比Ant更简单、更先进 |

# 关键技术的解决方案

## Struts2框架

Apache Struts2即是之前大家所熟知的WebWork2。在经历了几年的各自发展后，WebWork和Struts社区决定合二为一，也即是Struts 2。Struts2是一个兼容Struts1和WebWork的MVC框架，它是以Webwork的设计思想为核心，吸收了Struts1的优点。



**Struts2体系结构**

Struts2框架的大致处理流程如下：

* 浏览器发送请求，例如请求/mypage.action、/reports/myreport.pdf等；
* 核心控制器FilterDispatcher根据请求调用合适的Action；
* WebWork的拦截器链自动对请求应用通用功能，例如workflow、validation或文件上传等功能；

回调Action的execute方法，该execute方法先获得用户请求参数，然后执行某种数据操作，既可以是将数据保存到数据库，也可以从数据库中检索数据。实际上Action只是一个控制器，他会调用业务逻辑组件来处理用户的请求。

**Struts1.x与Struts2比较**

|  |  |  |
| --- | --- | --- |
| **特性** | **Struts1.x** | **Struts2** |
| **Action类** | Struts1.x要求Action类要扩展自一个抽象基类。Struts1.x的一个共有的问题是面向抽象类编程而不是面向接口编程。 | Struts2的Action类实现了一个Action接口，连同其他接口一起来实现可选择和自定义的服务。Struts2提供一个名叫ActionSupport的基类来实现一般使用的接口。当然，Action接口不是必须的。任何使用execute方法的POJO对象可以被当作Struts 2的Action对象来使用。 |
| **线程模型** | Struts1.x Action类是单例类，因为只有一个实例来控制所有的请求。单例类策略造成了一定的限制，并且给开发带来了额外的烦恼。Action资源必须是线程安全或者同步的。 | Struts2 Action对象为每一个请求都实例化对象，所以没有线程安全的问题。（实践中，servlet容器给每一个请求产生许多丟弃的对象，并且不会导致性能和垃圾回收问题）。 |
| **Servlet 依赖** | Struts1.x的Action类依赖于servlet API，当Action被调用时，以HttpServletRequest和HttpServletResponse作为参数传给execute方法。 | Struts2的Action和容器无关。Servlet上下文被表现为简单的Maps，允许Action被独立的测试。Struts2的Action可以访问最初的请求(如果需要的话)。但是，尽可能避免或排除其他元素直接访问HttpServletRequest或HttpServletResponse。 |
| **易测性** | 测试Struts1.x的主要问题是execute方法暴露了Servlet API这使得测试要依赖于容器）。第三方的扩展，如Struts TestCase，提供了一套Struts1的模拟对象（来进行测试）。 | Struts2的Action可以通过初始化、设置属性、调用方法来测试。依赖注入的支持也是测试变得更简单。 |
| **捕获输入** | Struts1.x使用ActionForm对象来捕获输入。象Action一样，所有的ActionForm必须扩展基类。因为其他的JavaBean不能作为ActionForm使用，开发者经常创建多余的类来捕获输入。DynaBeans可以被用来作为替代ActionForm的类来创建。但是，开发者可能是在重新描述(创建)已经存在的JavaBean（仍然会导致有冗余的javabean）。 | Struts2直接使用Action属性作为输入属性，消除了对第二个输入对象的需求。输入属性可能是有自己(子)属性的rich对象类型。Action属性能够通过web页面上的taglibs访问。Struts2也支持ActionForm模式。rich对象类型，包括业务对象，能够用作输入/输出对象。这种ModelDriven 特性简化了taglib对POJO输入对象的引用。 |
| **表达式语言** | Struts1.x整合JSTL，所以它使用JSTL的表达式语言。表达式语言有基本的图形对象移动，但是对集合和索引属性的支持很弱。 | Struts2使用JSTL，但是也支持一个更强大和灵活的表达式语言－－"Object Graph Notation Language" (OGNL)。 |
| **将值绑定到页面** | Struts1.x使用标准JSP机制来绑定对象到页面上下文。 | Struts2使用“ValueStack”技术，使taglib能够访问值而不需要把你的页面（view）和对象绑定起来。ValueStack策略允许通过一系列名称相同但类型不同的属性重用页面（view）。 |
| **类型转换** | Struts1.x的ActionForm属性经常都是String。Struts 1.x使用Commons-Beanutils来进行类型转换。转换每一个类，而不是为每一个实例配置。 | Struts2使用OGNL进行类型转换。提供基本和常用对象的转换器。 |
| **验证** | Struts1.x支持在ActionForm的validate方法中手动校验，或者通过Commons Validator的扩展来校验。同一个类可以有不同的校验内容，但不能校验子对象。 | Struts2支持通过validate方法和XWork校验框架来进行校验。XWork校验框架使用为属性类类型定义的校验和内容校验，来支持chain校验子属性 |
| **Action执行控制** | Struts1.x支持每一个模块有单独的Request Processors（生命周期），但是模块中的所有Action必须共享相同的生命周期。 | Struts2支持通过拦截器堆栈（Interceptor Stacks）为每一个Action创建不同的生命周期。堆栈能够根据需要和不同的Action一起使用。 |

**总结：**

根据struts1.x和struts2.0的对比，struts2.0提供的方法更灵活更易于开发，所以决定采用struts2.0作为教学机管理系统2.0的前台框架。

## 持久层框架

持久层框架目前比较流行的有Hibernate、ibatis等，下面主要对这两种框架进行介绍。

**Hibernate**

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。

一、Hibernate是JDBC 的轻量级的对象封装，它是一个独立的对象持久层框架，和App Server，和EJB没有什么必然的联系。Hibernate可以用在任何JDBC可以使用的场合，例如Java应用程序的数据库访问代码，DAO接口的实现类，甚至可以是BMP里面的访问数据库的代码。从这个意义上来说，Hibernate和EB不是一个范畴的东西，也不存在非此即彼的关系。

二、Hibernate是一个和JDBC密切关联的框架，所以Hibernate的兼容性和JDBC驱动，和数据库都有一定的关系，但是和使用它的Java程序，和App Server没有任何关系，也不存在兼容性问题。

三、 Hibernate不能用来直接和Entity Bean做对比，只有放在整个J2EE项目的框架中才能比较。并且即使是放在软件整体框架中来看，Hibernate也是做为JDBC的替代者出现的，而不是Entity Bean的替代者出现的。

传统的架构：

1）Session Bean <-> Entity Bean <-> DB

为了解决性能障碍的替代架构：

2）Session Bean <-> DAO <-> JDBC <-> DB

使用Hibernate来提高上面架构的开发效率的架构：

3）Session Bean <-> DAO <-> Hibernate <-> DB

就上面3个架构来分析：

* 内存消耗：采用JDBC的架构2无疑是最省内存的，Hibernate的架构3次之，EB的架构1最差。
* 运行效率：如果JDBC的代码写的非常优化，那么JDBC架构运行效率最高，但是实际项目中，这一点几乎做不到，这需要程序员非常精通JDBC，运用 Batch语句，调整PreapredStatement的Batch Size和Fetch Size等参数，以及在必要的情况下采用结果集cache等等。而一般情况下程序员是做不到这一点的。因此Hibernate架构表现出最快的运行效率。 EB的架构效率会差的很远。
* 开发效率：在有JBuilder的支持下以及简单的项目，EB架构开发效率最高，JDBC次之，Hibernate最差。但是在大的项目，特别是持久层关系映射很复杂的情况下，Hibernate效率高的惊人，JDBC次之，而EB架构很可能会失败。
* 分布式，安全检查，集群，负载均衡的支持 由于有SB做为Facade，3个架构没有区别。

**Ibatis**

使用ibatis 提供的ORM机制，对业务逻辑实现人员而言，面对的是纯粹的Java对象， 这一层与通过Hibernate 实现ORM 而言基本一致，而对于具体的数据操作，Hibernate 会自动生成SQL 语句，而ibatis 则要求开发者编写具体的SQL 语句。相对Hibernate等 “全自动”ORM机制而言，ibatis 以SQL开发的工作量和数据库移植性上的让步，为系统 设计提供了更大的自由空间。

**Hibernate与ibatis优缺点比较：**

1． iBATIS非常简单易学，Hibernate相对较复杂，门槛较高。

2． 二者都是比较优秀的开源产品

3． 当系统属于二次开发,无法对数据库结构做到控制和修改,那iBATIS的灵活性将比Hibernate更适合

4． 系统数据处理量巨大，性能要求极为苛刻，这往往意味着我们必须通过经过高度优化的SQL语句（或存储过程）才能达到系统性能设计指标。在这种情况下iBATIS会有更好的可控性和表现。

5． iBATIS需要手写sql语句，也可以生成一部分，Hibernate则基本上可以自动生成，偶尔会写一些Hql。同样的需求,iBATIS的工作量比Hibernate要大很多。类似的，如果涉及到数据库字段的修改，Hibernate修改的地方很少，而iBATIS要把那些sql mapping的地方一一修改。

6． 以数据库字段一一对应映射得到的PO和Hibernte这种对象化映射得到的PO是截然不同的，本质区别在于这种PO是扁平化的，不像Hibernate映射的PO是可以表达立体的对象继承，聚合等等关系的，这将会直接影响到你的整个软件系统的设计思路。

7． Hibernate现在已经是主流O/R Mapping框架，从文档的丰富性，产品的完善性，版本的开发速度都要强于iBATIS

**总结：**

根据本项目对sql优化灵活性的要求，选择ibatis作为数据持久层框架。

## Ajax技术

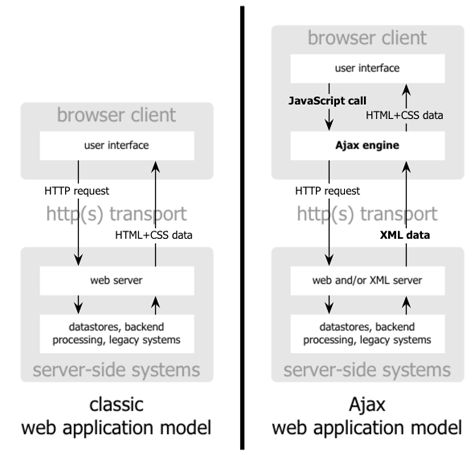
**Ajax概述：**

Ajax不是一种技术。实际上，它由几种蓬勃发展的技术以新的强大方式组合而成。Ajax包含：

* XHTML和CSS
* 使用文档对象模型(Document Object Model)作动态显示和交互
* 使用XML和XSLT做数据交互和操作
* 使用XMLHttpRequest进行异步数据接收
* 使用JavaScript将它们绑定在一起

　　传统的web应用模型工作起来就象这样:大部分界面上的用户动作触发一个连接到Web服务器的HTTP请求。

　　服务器完成一些处理---接收数据，处理计算，再访问其它的数据库系统，最后返回一个HTML页面到客户端。这是一个老套的模式，自采用超文本作为web使用以来，一直都这样用, 但看过《The Elements of User Experience》的读者一定知道，是什么限制了Web界面没有桌面软件那么好用。



**传统Web应用模型(左)与Ajax模型的比较(右).**

　　这种旧的途径让我们认识到了许多技术，但它不会产生很好的用户体验。当服务器正在处理自己的事情的时候，用户在做什么?没错，等待。每一个动作，用户都要等待。

　　很明显，如果我们按桌面程序的思维设计Web应用，我们不愿意让用户总是等待。当界面加载后，为什么还要让用户每次再花一半的时间从服务取数据?实际上，为什么老是让用户看到程序去服务器取数据呢?

　　Ajax如何不同凡响

　　通过在用户和服务器之间引入一个Ajax引擎，可以消除Web的开始-停止-开始-停止这样的交互过程. 它就像增加了一层机制到程序中，使它响应更灵敏，而它的确做到了这一点。

不像加载一个页面一样，在会话的开始，浏览器加载了一个Ajax引擎---采用JavaScript编写并且通常在一个隐藏frame中。这个引擎负责绘制用户界面以及与服务器端通讯。Ajax引擎允许用异步的方式实现用户与程序的交互--不用等待服务器的通讯。所以用户再不不用打开一个空白窗口，看到等待光标不断的转，等待服务器完成后再响应。

使用Ajax的最大优点，就是能在不更新整个页面的前提下维护数据。这使得Web应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

传统的web应用允许用户填写表单(form)，当提交表单时就向[web](http://baike.baidu.com/view/3912.htm)[服务器](http://baike.baidu.com/view/899.htm)发送一个请求。服务器接收并处理传来的表单，然後返回一个新的[网页](http://baike.baidu.com/view/828.htm)。这个做法浪费了许多带宽，因为在前後两个页面中的大部分HTML代码往往是相同的。由于每次应用的交互都需要向服务器发送请求，应用的响应时间就依赖于服务器的[响应时间](http://baike.baidu.com/view/84619.htm)。这导致了用户界面的响应比本地应用慢得多。

与此不同，AJAX应用可以仅向服务器发送并取回必需的数据，它使用SOAP或其它一些基于XML的web service接口，并在客户端采用JavaScript处理来自服务器的响应。因为在服务器和浏览器之间交换的数据大量减少，结果我们就能看到响应更快的应用。同时很多的处理工作可以在发出请求的客户端机器上完成，所以Web服务器的处理时间也减少了。

Ajax应用程序的优势在于：

* 通过异步模式，提升了用户体验
* 优化了[浏览器](http://baike.baidu.com/view/7718.htm)和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
* Ajax引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载。

**Ajax开发框架：**

毫无疑问，Ajax作为当前最火爆的技术之一，其优秀的框架层出不穷。Prototype、Dwr、Dojo、JQuery、YUi…… 都是非常出色的产品。

1. **JQuery**

　　特点：短小精悍(19k)，接口设计得精妙(自然语言的风格)，与程序思路配合精密。极大限度地体现了javascript的特性；支持xpath查询，dom1-3，轻松选择需要的元素；css支持；简单的动画实现，支持自定义动画；支持插件开发，现有插件多；完整的api文档以及范例，易学；拥有官方UI程序供使用，效果好。

1. **Yui**

　　特点：Yahoo发布的AJAX组件库，是一个包含了各个方面，从工具类库到通讯，到UI组件的综合性JS库。YUL的最大优势在于文档非常齐全，而且有Yahoo的支持，缺点是库目前还不全，功能也不强大。

1. **Ext**

　　 特点：Ext来自于对YUI的扩展，扩展後功能和界面都有了很大的提高。初期仅仅是对YUI的对话框扩展，后来逐渐有了自己的特色，深受网友的喜爱。Ext封装了很多组件用于UI的展示，Ext的所有组件都是扩展于Ext.Component, 而后子类扩展和集成形成了一个单根的组件树.

1. **Prototype**

　　特点：一个非常优雅的JS库，定义了JS的面向对象扩展，DOM操作API，事件等等，之上还有rico/script.aculo.us实现一些JS组件功能和效果(尚不够完善)，以prototype为核心，形成了一个外围的各种各样的JS扩展库，是相当有前途的JS底层框架，突出特点就是非常易学易用，门槛很低，常常是一两行JS代码就可以搞定一个相关的功能。同时它也是RoR集成的AJAX JS库。

1. **Dojo**

　　特点：Dojo包括了Javascript本身的语言扩展，以及各个方面的工具类库，和比较完善的UI组件库；Dojo设计的包加载机制(Package System)和模块化(Libraries)的结构，能保持更好的扩展性，提高执行性能，减轻了用户开发的工作量，并保持一定的灵活性(用户可以自己编写扩展)；Dojo官方网站有着丰富的学习资源；专业的开发团队，可以保证更新速度及质量。

1. **Mootools**

　　特点：小巧高效，完整下载36k；模块化设计，合理规范，优雅的OOP风格；创新的下载过程，可以跟据自己的需要勾选相应的模块下载，Build Your Own Framework；Effects模块(moo.fx)轻量高效,可以实现优雅、可定制、easing的动画；完整的API文档，丰富的范例。

1. **Dwr**

　　特点：把java类转化为javascript类由dwr自动完成，只需简单的配置；应用起来极其简单。开发者不要该服务器代码就可以集成；容易测试。和webwork一样，隐藏的http协议；强扩展性。例如与spring集成，只需修改一点代码；性能。就与jason等简单比较，dwr性能可能是最好的。

1. **Buffalo**

　　特点：国人开发的Ajax框架。定义了Web远程调用的传输基础，并且将远程调用对象完整的序列化到了本地，成为可以被JavaScript编程触及的对象。Buffalo中的重要组件-BuffaloBinding，提供了将JavaScript对象绑定到HTML元素的能力。这种绑定将是无侵入的，只需要在HTML元素中加入若干个不影响排版的属性，即可将数据与界面绑定。

1. **Qooxdoo**

　　特点：不通过常规的HTML来构造页面，完全使用JS以类似VB/Delphi风格的编程方式构造Web GUI界面，比较适合内网面向C/S风格的web应用，而不适合面向Internet的界面多变风格的应用。

1. **Spry**

特点：设计规范，功能全面，文档丰富,面向设计人员而不是开发人员。与其它一些Ajax框架相比，它的服务器端的技术不是很可靠。它依赖于XML，XML可以很容易被Spry组件接受。

总结：

根据本项目对页面性能要求，要选择性能高的框架jquery作为页面js库，同时选择ext作为皮肤js库。

## XML解析器

目前流行的XML解析器主要有DOM、SAX、JDOM、DOM4J等，下面我们就这四种解析器进行分析和比较：

1. **DOM**

　　DOM 是用与平台和语言无关的方式表示 XML 文档的官方 W3C 标准。DOM 是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。(所以其劣势就是基与大文件的加载速度很慢,因为它是需要全部加载后才操作的).

　　用 DOM 解析模型的优点是编程容易，开发人员只需要调用建树的指令，然后利用navigation APIs访问所需的树节点来完成任务。可以很容易的添加和修改树中的元素。然而由于使用 DOM 解析器的时候需要处理整个 xml(标准化越来越近了) 文档，所以对性能和内存的要求比较高，尤其是遇到很大的 xml(标准化越来越近了) 文件的时候。由于它的遍历能力，DOM 解析器常用于 xml(标准化越来越近了) 文档需要频繁的改变的服务中。

　　另一方面，对于特别大的文档，解析和加载整个文档可能很慢且很耗资源，因此使用其他手段来处理这样的数据会更好。这些基于事件的模型，比如 SAX。

1. **SAX**

SAX 解析器采用了基于事件的模型，它在解析 xml(标准化越来越近了) 文档的时候可以触发一系列的事件，当发现给定的tag的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。这种处理的优点非常类似于流媒体的优点。分析能够立即开始，而不是等待所有的数据被处理。而且，由于应用程序只是在读取数据时检查数据，因此不需要将数据存储在内存中，对内存的要求通常会比较低。这对于大型文档来说是个巨大的优点。事实上，应用程序甚至不必解析整个文档;它可以在某个条件得到满足时停止解析。特别是当开发人员只需要处理文档中所包含的部分数据时，SAX 这种扩展能力得到了更好的体现。但用 SAX 解析器的时候编码工作会比较困难，而且很难同时访问同一个文档中的多处不同数据。

1. **JDOM**

　　JDOM的目的是成为 Java 特定文档模型，它简化与 XML 的交互并且比使用 DOM 实现更快。

　　JDOM 与 DOM 主要有两方面不同。首先，JDOM 仅使用具体类而不使用接口。这在某些方面简化了 API，但是也限制了灵活性。第二，API 大量使用了 Collections 类，简化了那些已经熟悉这些类的 Java 开发者的使用。

　　JDOM 文档声明其目的是“使用 20%(或更少)的精力解决 80%(或更多)Java/XML 问题”(根据学习曲线假定为 20%)。JDOM 对于大多数 Java/XML 应用程序来说当然是有用的，并且大多数开发者发现 API 比 DOM 容易理解得多。JDOM 还包括对程序行为的相当广泛检查以防止用户做任何在 XML 中无意义的事。然而，它仍需要您充分理解 XML 以便做一些超出基本的工作(或者甚至理解某些情况下的错误)。这也许是比学习 DOM 或 JDOM 接口都更有意义的工作。

JDOM 自身不包含解析器。它通常使用 SAX2 解析器来解析和验证输入 XML 文档(尽管它还可以将以前构造的 DOM 表示作为输入)。它包含一些转换器以将 JDOM 表示输出成 SAX2 事件流、DOM 模型或 XML 文本文档。JDOM 是在 Apache 许可证变体下发布的开放源码。

1. **DOM4J**

虽然 DOM4J 代表了完全独立的开发结果，但最初，它是 JDOM 的一种智能分支。它合并了许多超出基本 XML 文档表示的功能，包括集成的 XPath 支持、XML Schema 支持以及用于大文档或流化文档的基于事件的处理。它还提供了构建文档表示的选项，它通过 DOM4J API 和标准 DOM 接口具有并行访问功能。从 2000 下半年开始，它就一直处于开发之中。

为支持所有这些功能，DOM4J 使用接口和抽象基本类方法。DOM4J 大量使用了 API 中的 Collections 类，但是在许多情况下，它还提供一些替代方法以允许更好的性能或更直接的编码方法。直接好处是，虽然 DOM4J 付出了更复杂的 API 的代价，但是它提供了比 JDOM 大得多的灵活性。

在添加灵活性、XPath 集成和对大文档处理的目标时，DOM4J 的目标与 JDOM 是一样的：针对 Java 开发者的易用性和直观操作。它还致力于成为比 JDOM 更完整的解决方案，实现在本质上处理所有 Java/XML 问题的目标。在完成该目标时，它比 JDOM 更少强调防止不正确的应用程序行为。

1. **总结**

DOM是基于平台、语言无关的官方W3C标准。基于树的层次，其优点是可以移植，编程容易，开发人员只需要调用建树的指令。

JDOM是想成为 Java 特定文档模型。它简化与 XML 的交互并且比使用 DOM 实现更快。使用的是具体类不使用接口，运用了大量的COllections类，方便程序员。

JDOM 和 DOM 在性能测试时表现不佳，在测试 10M 文档时内存溢出。在小文档情况下还值得考虑使用 DOM 和 JDOM。

　　SAX是基于事件模型的。它在解析 XML 文档的时候可以触发一系列的事件，当发现给定的tag的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。类似与流媒体的解析方式，所以在加载大文件时效果不错。其缺点是SAX 解析器的时候编码工作会比较困难，而且很难同时访问同一个文档中的多处不同数据。

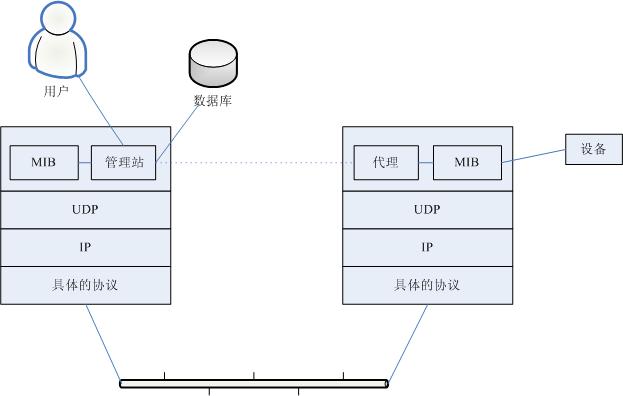
　　DOM4J 是一个非常非常优秀的Java XML API，具有性能优异、功能强大和极端易用使用的特点，同时它也是一个开放源代码的软件。如今你可以看到越来越多的 Java 软件都在使用 DOM4J 来读写 XML，特别值得一提的是连 Sun 的 JAXM 也在用 DOM4J。Hibernate 也用 DOM4J 来读取 XML 配置文件。如果不考虑可移植性，推荐采用DOM4J。

## SNMP

简单网络管理协议(SNMP)首先是由Internet工程任务组织(Internet Engineering Task Force)(IETF)的研究小组为了解决Internet上的路由器管理问题而提出的。它可以在IP，IPX，AppleTalk，OSI以及其他用到的传输协议上被使用。

简单网络管理协议(SNMP)是最早提出的网络管理协议之一，它一推出就得到了广泛的应用和支持，特别是很快得到了数百家厂商的支持，其中包括IBM，HP，SUN等大公司和厂商。目前SNMP已成为网络管理领域中事实上的工业标准，并被广泛支持和应用，大多数网络管理系统和平台都是基于SNMP的。

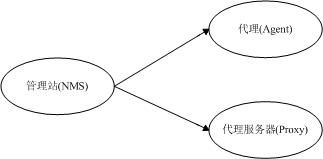
具体实现上，SNMP为管理员提供一个网管平台，负责网管命令的发出与数据的存储及分析等。被监管的设备上则运行一个SNMP代理，代理实现设备与管理站的SNMP通讯。

  
图3：管理站与代理的通讯

管理站和代理端使用MIB进行接口统一，MIB定义了设备中被管理对象，管理站和代理都实现相应的MIB对象，使得双方可以识别对方的数据，实现通讯。管理站向代理申请MIB中定义的数据，代理识别后，将管理设备提供的相关状态或参数等数据转换成MIB定义的格式，应答给管理站，完成一次管理操作。

SNMP的这种结构过程，使得网络设备对SNMP的支持极为方便。已有的设备，只要新加一个SNMP模块就可以实现网管。旧的带扩展槽的设备，新实现一个SNMP模块插上即可支持网管。网络上的许多设备，比如：路由器、交换机等，都可以通过添加一个SNMP网管模块增加网管功能。服务器可通过运行一个网管进程实现。其它服务级的产品也可以提供网管模块实现网络管理，比如Oracle、Weblogic都有snmp进程，运行后就可以通过网络管理对这些系统级服务进行管理。

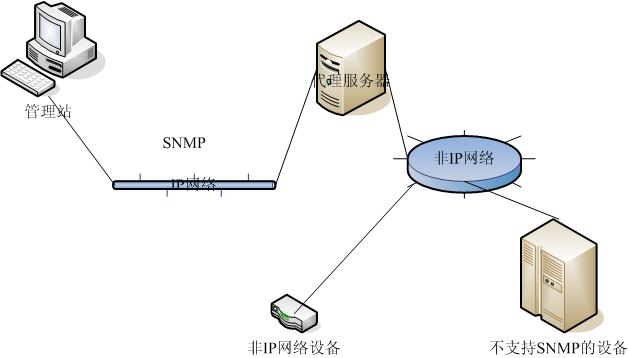
SNMP系统中定义3个重要的角色：网络管理系统（NMS又名网络管理站）、代理（Agent）、代理服务器（Proxy）。

  
图4：SNMP中的角色

        网络管理系统（管理站）是系统总控台，向管理员提供界面以获取与改变设备的配置、信息、状态、操作等。管理站与代理(Agent)进行通讯，执行相应的Set与Get操作，以读取或设置设备配置，并接收代理发过来的警报（Trap）。

        代理（Agent）介于管理站与管理信息库（MIB）之间，与管理站通讯并响应管理站的请求，从MIB获取或设置相应数据，以响应响应的请求，或根据MIB的相应数据决定Trap的发送。

        代理服务器（Proxy）是一种特殊的代理，在不能直接使用SNMP协议的地方，比如异种的网络、不同版本SNMP代理等情况下，代理代替相关设备向管理站提供一种外观，为设备代理SNMP协议的实现。

  
图5：代理服务器的典型应用

除却这3个角色，SNMP还包含两个重要的部分：

       管理信息库（MIB）定义了设备可使用的管理信息，由设备维护，以供Agent使用。

比如，设备信息组定义了这样一些管理对象，用来描述设备的基本信息：

|  |  |  |  |
| --- | --- | --- | --- |
| **名 称** | **数据类型** | **R/W** | **描 述** |
| sysDescr | DisplayString |  | 设备的文字描述，对设备硬件、操作系统等的说明 |
| sysObjectID | ObjectID |  | 设备包含的网管子系统的供应商授权身份 |
| sysUpTime | TimeTicks |  | 从系统的网管系统启动以来运行的时间（百分之一秒） |
| sysContact | DisplayString | \* | 设备联系人的名字及联系方式 |
| sysName | DisplayString | \* | 在管理上被赋予的名称 |
| sysLocation | DisplayString | \* | 结点的物理位置 |
| sysServices | DisplayString  [0…127] | \* | 指示结点提供的服务的值。该值是此结点所支持的OSI模型中层次的和。根据所提供的服务，将下面的一些值相加：0x01（物理层）、0x02（数据链路层）、0x04（互联网层）、0x08（端到端的运输层）、0x40（应用层） |

       被管理的设备是SNMP系统中的被管理对象。设备提供MIB需要的数据，将自身的各种数据登记到MIB中，供SNMP实体使用。

SNMP中主要提供了五类管理操作：get操作用来提取特定的网络管理信息；get-next操作通过遍历活动来提供强大的管理信息提取能力；get-bulk操作可以用于批量获取数据。set操作用来对管理信息进行控制(修改、设置)；trap操作用来报告重要的事件。

SNMP4J是一个使用JAVA开发的开源的SNMP框架。它实现了对SNMP协议的封装，提供了SNMP管理者和Agent的基础实现，支持SNMP v1、v2c和v3等协议，开发者可以在它的基础上构建自己的系统。网管系统2.x版本采用了这个框架。

**总结：**

由于snmp的标准性，本系统采用snmp作为采集设备性能指标的基础协议，方便系统基础数据采集的功能扩展。

.

# 系统架构解决方案

## 架构1



## 架构2



# 本项目确定的方案

根据本项目的实际需求，结合以上各种技术解决方案的特点，确定了教学机管理系统的技术方案：

## 系统架构

方案1与方案2的区别在于方案1比方案2多了管理服务中间层，所以在系统设计上会有如下不同点：

1、方案1需要设计管理服务与管理平台、agent的接口，而方案2则不需要设计与管理平台的接口。

2、方案1将管理服务当成一个子系统进行设计，而方案2将管理服务的功能当成一个系统内在模块进行设计。

分析：

1. 从系统架构的可维护性来说方案1比方案2更优越。

如果教学机管理系统以学校为单位进行部署，方案2也可以满足要求，考虑到随着市场的发展，如果要建立一个比学校大的管理平台就必须需要管理服务中间层承担数据采集、接收处理的任务，所以专门设立管理服层负责与agent接口进行数据采集接收处理，这样很好的对系统进行了分层，非常有利于将来系统的扩展和维护。

1. 从开发工作量来说，方案2比方案1少了与管理平台接口的开发。

由于方案2少了与管理平台接口所以会比方案1的开发工作量少些，但存在一个问题是方案2将管理服务的功能设计成为一个模块势必会与管理平台有很多的共享数据存在，这样是不利于多人协作开发的。

所以，教学机管理系统2.0采用方案1为系统架构。

## 技术方案

* + - 管理层采用Struts2+Spring2作为底层框架；页面展示采用Ext框架；Ajax交互采用JQuery框架；数据的持久化采用iBatis框架；网络拓扑图采用Flash技术。
    - 服务访问接口提供WebService、Socket等对外接口方式，管理平台访问应用服务采用Rmi-IIop。
    - 应用服务层采用Spring2作为基础框架，主要应用它的任务调度等技术进行各种采集任务的调度；采用snmp4J框架进行snmp交互，采用JRobin来进行实时性能数据的存取、绘制性能分析图形。
    - 采用C语言开发终端管理服务软件。
    - 采用Log4j统一进行日志处理。
    - 使用JUnit进行单元测试。