系统分析与设计

**科技专家资源共享平台**

**14-04-架构设计**

|  |  |  |
| --- | --- | --- |
| **项目组成员信息** | | |
| **小组名称** | **那么队名就只有一个了！** | |
| **小组联系人** | **李光昊** | |
| **学号** | **姓名** | **本文档中主要承担的工作内容** |
| 16211058 | 李光昊 | 总文档整合与格式排版 |
| 16211059 | 孟庆博 | 切面关注点与爬虫技术 |
| 16211061 | 闫晨雨 | 部署策略与技术选型指导 |
| 16211068 | 曾骏鹏 | 技术选型 |
| 16211067 | 陈喆 | 切面关注点 |
| 16211065 | 乔禹 | 技术选型 |
| 16211057 | 邹健伦 | 部署策略 |
| 16211060 | 张子旺 | 质量属性 |
| 16211072 | 刘铭辉 | 质量属性 |

2019-04

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0 | 2019.4.12 | 李光昊 | 李光昊 | 文档整合排版 |

目 录

[引言 1](#_Toc5993958)

[一、应用类型 1](#_Toc5993959)

[二、部署策略 1](#_Toc5993960)

[2.1 MVC模型 1](#_Toc5993961)

[2.2 设备部署结构 2](#_Toc5993962)

[2.3 管理软件选择 3](#_Toc5993963)

[三、技术选型 3](#_Toc5993964)

[四、质量属性 6](#_Toc5993965)

[4.1 开发期质量属性 6](#_Toc5993966)

[4.1.1 概念完整性 6](#_Toc5993967)

[4.1.2 可维护性 6](#_Toc5993968)

[4.1.3 可重用性 6](#_Toc5993969)

[4.2 运行期质量属性 6](#_Toc5993970)

[4.2.1 性能效率 7](#_Toc5993971)

[4.2.2 可伸缩性 7](#_Toc5993972)

[4.2.3 互操作性 7](#_Toc5993973)

[4.2.4 可靠性 7](#_Toc5993974)

[4.3 系统质量属性 7](#_Toc5993975)

[4.3.1 可支持性 7](#_Toc5993976)

[4.3.2 可测试性 8](#_Toc5993977)

[4.4 用户使用质量属性 8](#_Toc5993978)

[五、切面关注点 8](#_Toc5993979)

[5.1 安全模块 8](#_Toc5993980)

[5.1.1 Java语言安全性和字节码验证 8](#_Toc5993981)

[5.1.2 加解密方式 8](#_Toc5993982)

[5.1.3 认证 9](#_Toc5993983)

[5.1.4 安全通信 9](#_Toc5993984)

[5.2 缓存模块 9](#_Toc5993985)

[5.2.1 本地缓存 9](#_Toc5993986)

[5.2.2 分布式缓存 10](#_Toc5993987)

[5.3 信息模块 10](#_Toc5993988)

[总结 10](#_Toc5993989)

引言

在分析模型之后，我们的项目进入了架构设计部分。在面向对象的方法论中，设计是分析的自然延续，是对分析模型的进一步细化。这里强调的更多是所使用的的方法、工具和成果。分析的目标是明确做什么，而设计的目标则是讨论怎么做的问题。具体来说，分析重点关注业务问题本身，确定了将要构建的内容；而设计则关注系统的技术和实现细节。

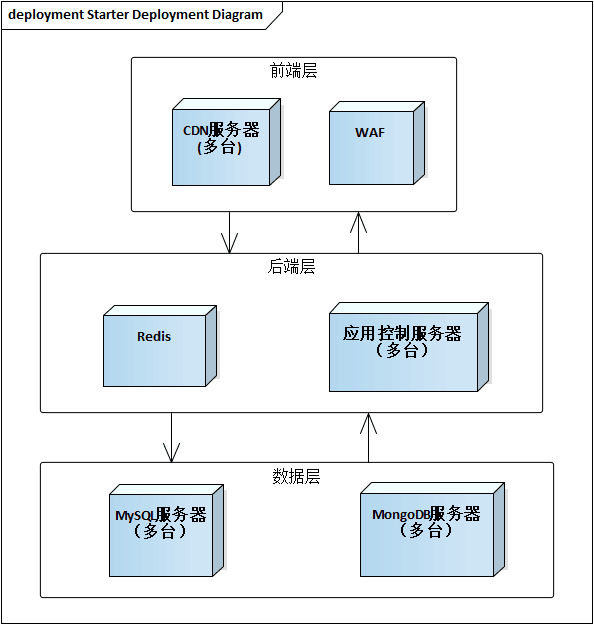
我们的架构设计分为应用类型、技术选型、部署策略、质量属性、切面关注点五个部分进行。

一、应用类型

科技专家资源共享平台的功能和中国知网、百度学术、谷歌学术等平台类似，而这些平台最初都是以网站的形式呈现的。作为检索科技成果资源与专家资源的平台，面向的用户一般会使用电脑进行科研工作及论文撰写，因而我们将平台的应用类型定位为网站，可以通过PC端或移动端的浏览器访问。

二、部署策略

2.1 MVC模型



**图2-1 MVC模型**

使用MVC模型分为前端（展示）层，后端（逻辑）层，和数据层。

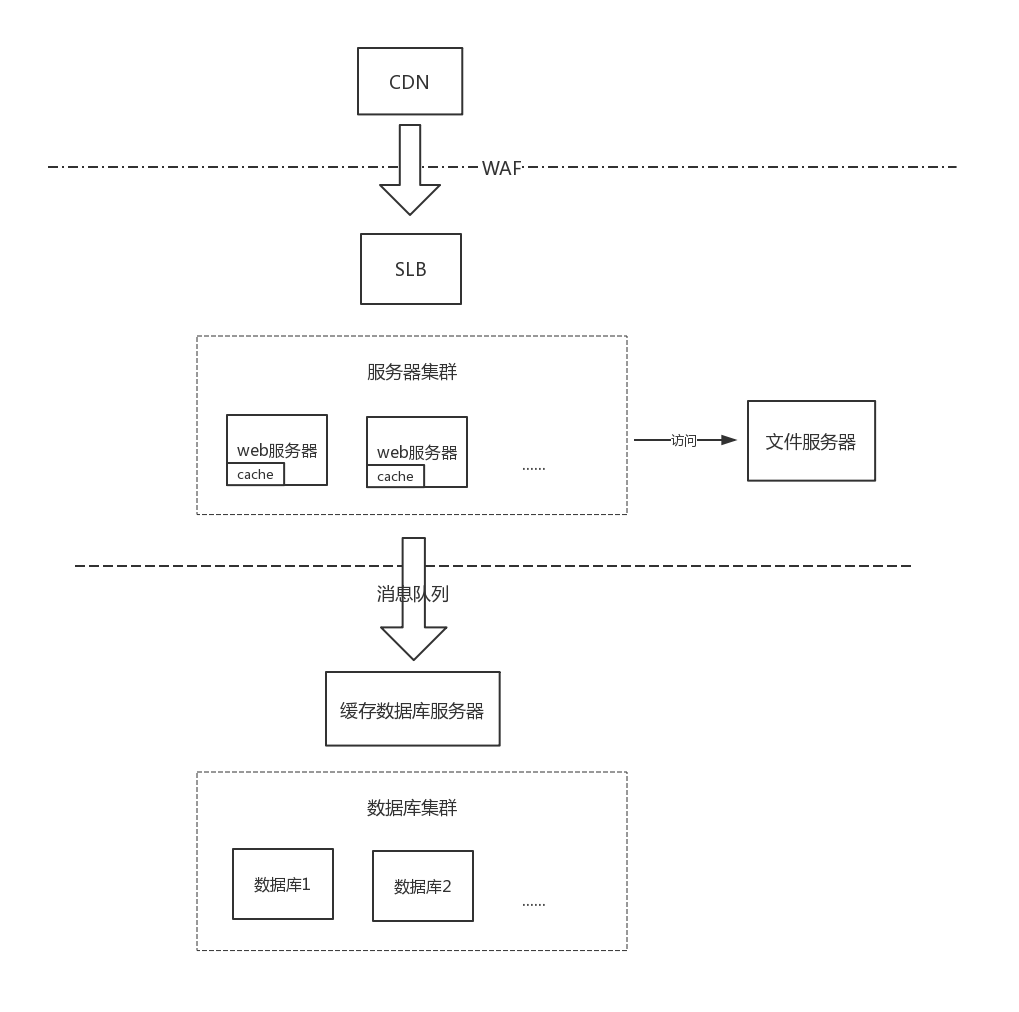
前端采用多台CDN服务器，将用户的访问指向距离最近的工作正常的缓存服务器上，由缓存服务器直接响应用户请求。前端与后端的通信通过WAF过滤，提高项目安全性。前端框架使用Vue.js。

后端层部署多台应用控制服务器（具体数量依据实际访问量设置），此外后端层还部署有Redis缓存，提高服务器的性能。后端框架采用Spring Boot。

数据层分为两种数据库，分别是存放论文，专利，专家等基本内容的MySQL和存放日志文件及实体关系（如论文和专家的对应关系）的MongoDB（NoSQL）。

服务器间的信息通讯使用消息队列实现。通过消息队列的异步处理，减少请求响应时间和解耦。(rabbitMQ/kafka)

2.2 设备部署结构



**图2-2 部署结构**

因为这是一个面向全国的平台，因此会带来各种问题。首先面向全国各地，许多距离远的用户就会访问延迟高；其次会产生很高的并发访问量，这会带来两个问题，一个是服务器承载请求量的能力，一个是数据库并发访问的问题；第三会产生需要存储的数据量巨大的问题；最后要考虑到服务规模的扩展性，因为在很多大城市用户量可能会激增。

首先我们会用CDN把服务部署到全国各地，以此来减少全国各地用户访问平台时的延迟。我们会设置防火墙控制访问权限，并采用加密传输。

在具体城市，我们会使用web服务器集群，并通过SLB来负载均衡、管理服务器集群，以此来解决高并发访问量的问题。web服务器中有对数据库内容的缓存，这样就能减少查询数据库耗时的问题。对于缓存这里（两次缓存都是这样），用户如果是查询就不用多说，如果是写入的话，会在数据库写入完毕后立即反向更新各缓存中相关的数据。这听起来可能消耗很大，不过实际上写入操作实际上很少：发送消息时可能涉及到写入，不过相关人员很少，几乎不会涉及到缓存更新；管理员审核通过时可能会修改，不过这修改量很小，而且可以在用户访问量较小的时候进行写入操作。存储论文、专利文章，我们会使用专门的文件服务器来存储，并通过自己的web服务器请求文献数据。

对于数据库的高并发量请求问题，我们会通过消息队列的方式暂存数据库请求。缓存数据库服务器会搭载程序，多进程并发处理消息队列中请求。并且在该服务器上会有缓存数据库，来减少对最终数据库的查询。实际上，我们是采用两级缓存，两级服务器的方式来达到最终数据库的。

之后我们采用数据库集群，并采用合适的管理软件统一管理，来解决存储量庞大的问题。

最后是扩展性的问题，在上层服务器中，我们只需增加slb管理的服务器数量就能承担更多的请求量（甚至可以动态管理）。在下层中，缓存数据库服务器不一定是一个服务器，也可以用集群来实现（缓存的问题刚才已经提到），或者增大该服务器的内存，以此来增大消息队列的处理速率。因为从最开始我们就使用多级缓存，最终的集群数据库查询量已经被减小，而且我们选择的集群管理软件也有比较强大的并发承载能力，因此最终数据库的查询并非限制扩展的瓶颈。

2.3 管理软件选择

开始的cdn和slb服务我们是打算直接使用阿里云的服务，因为考虑到自己部署成本太高，而且阿里有淘宝这些高并发应用的经验，服务器有良好的承载能力。服务器直接使用阿里云的ecs，配套有相应的防火墙。

web服务器使用nginx，使用的缓存方案是EHcache，因为这种方案可能会占用很大的内存，在选择ECS服务器时可能会选择大内存的种类。文件服务器会部署专门的ftp服务器，并由内部的web服务器代理访问。（这也可能会消耗内存）

消息队列我们会使用kafka，缓存数据库服务器并没有web服务器，搭载特定的程序，并发地处理消息队列中的请求。在缓存数据库服务器上还会有redis数据库，用于缓存最终集群数据库的数据，统一由缓存数据库服务器上搭载的程序管理。

缓存数据库服务器的程序如果在redis中没有找到想要的数据，便会从最终的集群数据库中加载数据。集群数据库使用mysql cluster统一管理多个搭载mysql的数据库服务器。

三、技术选型

通过比较分析不同的技术，我们最终确定了我们的技术选型。具体内容如下。

表3-1 技术选型一览

|  |  |
| --- | --- |
| Application Infrastructure | JDK, JavaEE |
| Application Layer | Spring MVC |
| Collaboration/Integration/Workflow | Activiti |
| Data Access | JDBC |
| Database Server | MySQL, MongoDB |
| Development Tools | Eclipse, HBuilder |
| Mobile | Html5 |
| Rich Client | Swing, JavaFX |
| Rich Internet Application (RIA) | JavaScript |
| Services | Nginx |
| Web | JSP |
| Web Server | Tomcat, JBoss |

表3-2 Application Layer对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **框架语言** | **开发难度** | **开发效率** | **适用场景** |
| Django | Python | 集成度高，开发难度比较低 | 包含session/orm/中间件/admin，无需第三方插件 | 大中型应用 |
| flask | Python | 便于开发小项目 | 无ORM，admin | 小型应用开发 |
| Tornado | Python | 便于开发小项目 | 效率介于flask、django之间 | 长连接多的web应用 |
| Springboot | Java | 需要理解AOP感念，学习曲线较陡 | 支持orm，效率较高 | 适合大型项目 |
| Hibernate | Java | 易用，高效 | 批量数据处理存在不足 | 场景不复杂，性能要求不苛刻 |

表3-3 数据库选择

|  |  |
| --- | --- |
| MySQL | MongoDB |
| 关系型数据库 | 非关系型数据库 |
| 传统sql语句查询 | 文档型，可存储自述型数据 |
| 体系较成熟，应用广泛 | 快速，具有高扩展性 |
| 简单易用 | Failover机制 |
| 不支持热备份 | 没有事务机制 |

表3-4 Apache和Nginx的优缺点对比

|  |  |
| --- | --- |
| Apache | Nginx |
| 模块多 | 轻量级，占用更少的内存和资源 |
| 阻塞型 | 异步非阻塞 |
| bug相对少 | 负载均衡 |
| 稳定 | 配置简洁 |

表3-5 Swing和JavaFX对比

|  |  |
| --- | --- |
| Swing | JavaFX |
| 基于AWT，后续更新匮乏 | 新增了Group容器 |
| 无基本平台依赖性 | 支持FXML、CSS |
| 完全自绘控件 | 可移植到移动设备上 |

此外我们还对爬虫模块应采用的技术进行了设计。

1. 技术选择：想要批量获取大量的格式化的科研资源数据，就必须借助于网络爬虫。目前使用较多的爬虫技术主要依靠python语言，python语言可以方便地抓取网页，并且将其存入数据库。为了避免浪费太多时间在做重复的工作上，所以我们选择了使用Scrapy框架。Scrapy是一个为遍历爬行网站、分解获取数据而设计的应用程序框架，它可以应用在广泛领域：数据挖掘、信息处理和或者历史片（历史记录）打包等等。爬虫的程序结构很大程度上取决于网站的结构，所以不同的网站需要使用不同的爬虫程序。

2. 反爬机制：由于使用爬虫会加重网站服务器的负担，所以许多的网站都会禁止或限制爬虫。惯例上，网站会把自己禁止的爬虫写在网站的rebots.txt中。为了避免版权纠纷，我们在使用爬虫时会严格遵守网站的要求。有的网站会对用户的访问行为进行监视，限制同一IP的访问速度。所以我们采用了替身IP的技术，使用多个IP轮流访问。同时在一段时间的爬取之后，爬虫程序进行短暂的暂停，以避免被网站限速或者拒绝访问。

四、质量属性

4.1 开发期质量属性

4.1.1 概念完整性

4.1.1.1 统一的模块设计

后端根据不同服务划分模块，例如普通用户的服务与科技专家的服务可以分开。

4.1.1.2 统一的命名规范

1. 后端开发变量名、参数名与方法名称采用驼峰命名法，前端则采用下划线的命名法。

2. 类的命名采用帕斯卡命名法。命名尽量完整，可以清楚表达含义与作用。

4.1.1.3 统一的代码风格

1. 尽量避免一行过长的代码。

2. 适当空行，避免过多空行。

3. 花括号换行。

4.1.2 可维护性

可维护性其实说简单了就是代码的可被修改的容易程度。开发团队统一制定编码规范，每个人都必需按照编码规范进行编程。

适当的代码注释。每个类前应有代码注释段，说明class的主要用途，基本用法，注意事项；每次修改都需要在此处添加修改日志，修改日志尽量详细；每个方法前有注释来说明方法的用法，要说明方法功能，参数及返回值含义。

系统使用“高内聚、低耦合”的模块化设计。

4.1.3 可重用性

系统采用模块化设计，高内聚低耦合，方便模块代码的复用，节省时间，使逻辑更加清晰。

4.2 运行期质量属性

4.2.1 性能效率

考虑到用户等待时间过长或者响应时间过短可能引起的误操作，响应时间应控制在500-1000ms以内。

4.2.1.1 针对资源需求

1. 减少处理事件流所需的资源：提高计算效率、减少计算开销。

2. 减少处理事件的数量：管理事件率、控制采样频率。

3. 控制资源的使用：限制执行时间（如减少迭代次数）、限制队列大小。

4.2.1.2 针对资源管理

1. 引入并发：通过SLB来负载均衡、管理服务器集群，以此解决高并发访问量的问题；通过消息队列的方式暂存数据库请求，多进程并发处理消息队列中的请求，以此解决数据库的高并发量请求问题。

2. 维持数据或计算的多个副本：C/S结构中客户机C就是计算的副本，它能减少服务器计算的压力；高速缓存可以存放数据副本。

3. 增加可用资源：在成本允许时，尽量使用速度更快的处理器、内存和网络。

4.2.1.3 针对资源仲裁

通过如下调度策略实现资源的仲裁：先进先出FIFO、固定优先级调度、动态优先级调度、轮转调度、时限时间最早优先、静态调度、离线确定调度。

4.2.2 可伸缩性

在上层服务器中增加SLB管理的服务器数量就可以承担更多的请求量（动态管理），在下层服务器中，也可以根据需求通过服务器集群增大缓存数据库服务器的规模，以此增加消息队列的处理速率，另外SLB拥有比较强大的并发承载能力，以上策略为本系统提高了整体业务吞吐能力。

4.2.3 互操作性

本系统与第三方的支付系统、数据库等系统有密切关联，应定义标准的服务接口，并使用基于服务的架构。

定义标准的数据格式和语义保证系统内部的规范操作。

设计高内聚、低耦合的模块以获得最大的灵活性和可重用性。

4.2.4 可靠性

本系统应在一定时间内保证无故障运行，平均无故障时间不应低于7\*24 小时。

前后端均部署多台服务器，采用双机双工模式，两台服务器之间连有心跳通信链路，并有结构冗余（静态、动态）能力。

Ngnix基于反向代理的负载均衡。

采用N版本程序设计和防卫式程序设计两种容错技术，保证系统的高容错率。

4.3 系统质量属性

4.3.1 可支持性

日志系统记录系统运行中的交互信息以及系统的警告和异常信息。

4.3.2 可测试性

系统前后端分离，使用vue-cli进行模块化编程，确保每个模块之间耦合度尽可能低，便于分模块、分功能测试。

4.4 用户使用质量属性

1. 系统界面简洁清晰，功能分区明了，主要使用人群面向未登录用户和科技专家。

2. 系统的搜索功能可以精确匹配用户的搜索内容；用户可以通过资源评价来对资源进行选择，方便用户找到最好的资源和专家。

3. 用户可以在电脑也可以在手机上网页登录我们的系统,可以采用响应式网站设计，采用优秀的前端框架JQuery+ Bootstrap, 更好地适配不同的设备。

4. 简化用户的操作步骤，使系统利于使用。实用的操作界面，将关键的按钮或者信息用明显的文字或框架显示，提醒用户注意，当用户操作不当时弹出提示并且返回至正确操作界面如未登录直接返回登录界面。

五、切面关注点

切面关注点主要指贯穿整个系统的模块，被不同层级的模块所使用。在科技专家资源共享平台中，切面关注点主要包括安全模块、缓存模块和消息模块。

5.1 安全模块

安全性大致可分为运行环境安全和网站程序自身安全。

运行环境安全：保证系统机器及外设不受物理损坏；及时安装厂家（部署策略里有说明）发布的安全补丁，WAF直接承包给第三方云厂商（阿里云盾），nginx本身就有设计防攻击的模块只需要简单配置即可，加上阿里云ecs提供的配套防火墙，保证网络操作系统、web服务器系统及数据库系统本身的安全。

网站程序自身安全：全站添加防SQL注入、SQL盲注和防跨站脚本弱点等安全代码；通过服务器的安全防范管理以及定期网站安全监测和加固来预防网站运行期间的安全隐患；上线前的安全测试；Java语言的安全性，通过https协议加密保证通讯安全。

安全模块具体如下。

5.1.1 Java语言安全性和字节码验证

Java语言被设计为类型安全且易于使用。它提供了自动内存管理，垃圾收集和对数组的范围检查。Java中的字节码检查机制，可以检查内存管理违规行为，堆栈下溢或溢出以及非法数据类型转换。通过这些方面，可以使Java程序员能够更加专注于程序编写。

5.1.2 加解密方式

Java安全API主要在java.security.Provider类中，覆盖主要安全领域的API，包括密码学，公共密钥基础设施，认证，安全通信和访问控制。密码体系主要包括各种加密API：消息摘要算法、数字签名算法、对称批量加密、对称流加密、不对称加密、基于密码的加密（PBE）、椭圆曲线密码（ECC）、密钥协商算法、密钥生成器、消息认证码（MAC）、（伪）随机数发生器。

加解密接口是基于Provider的，允许多个互操作的密码实现。 Java平台还包含一个内置提供程序，它作为使用本地PKCS＃11（v2.x）的Token。 这个名为SunPKCS11的Provider允许Java应用程序无缝访问兼容于PKCS＃11的加密服务。

5.1.3 认证

认证是确定用户身份的过程。Java平台提供的API使应用程序可以通过可插入登录模块执行用户身份验证。 应用程序调用LoginContext类（在javax.security.auth.login包中），后者又引用一个配置。 该配置指定要使用哪个登录模块（javax.security.auth.spi.LoginModule接口的实现）来执行实际身份验证。

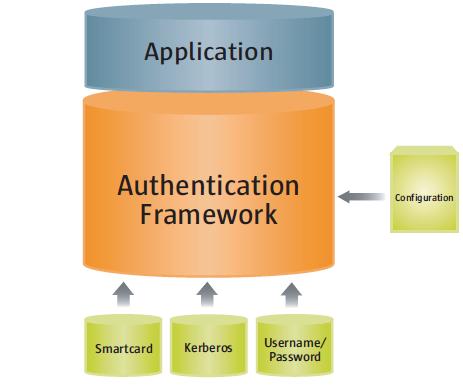


图5-1 认证

5.1.4 安全通信

通过网络传播的数据可以被不是预期收件人的人访问。当数据包括密码和信用卡号码等私人信息时，必须采取措施使数据对未授权方不可理解。Java平台提供SSL和TLS协议的API实现，其中包括数据加密，消息完整性，服务器身份验证和可选的客户端身份验证功能。

客户端身份认证主要使用SASL协议，SASL定义了如何交换认证数据，但是本身不指定数据的内容。这是一个框架，指定可用的认证数据的内容和语义的特定认证机制。 因特网社区为各种安全级别和部署方案定义了许多标准的SASL机制。

5.2 缓存模块

缓存主要包括本地缓存和分布式缓存两个模块。

5.2.1 本地缓存

在数据库和文件系统端，都已经存在现成的缓存系统，不需要进行设计。应用程序层的缓存主要在用户的磁盘中缓存一些图片或者UI元素，以避免每次访问时都需要从网站上下载，消耗大量的网络流量。对于缓存空间采用LRU策略，及时地将不常用的文件置换出去。当浏览器/或者客户端请求相应资源时，直接从本地缓存中取用。

5.2.2 分布式缓存

分布式的缓存需要使用 Ehcache框架，Ehcache是一个Java实现的开源分布式缓存框架，EhCache 可以有效地减轻数据库的负载，可以让数据保存在不同服务器的内存中，在需要数据的时候可以快速存取。同时EhCache 扩展非常简单，官方提供的Cache配置方式有好几种。你可以通过声明配置、在xml中配置、在程序里配置或者调用构造方法时传入不同的参数。

5.3 信息模块

信息的传递分为同层级和不同层级以及同步消息和异步消息。

对于同层级的信息传输，对象和对象之间可以直接使用方法调用的方式进行信息传递。对于不同层级的消息传递，在web端可以使用java中的request类传送简单类型的信息，也可以使用socket服务从服务端向客户端实时传递信息。

为了同时完成消息的同步和异步传输，对于web端，采用Java Web Service。WebService是一个以xml为基础的做消息传输的web工具，在微软平台有对应的.net实现的信息传输工具，在Java平台有JWS实现信息传输。WSDL是webservice的描述语言，使用xml描述对象（即webservice）的一切信息。SOAP为简单对象访问协议，通过这种协议，webservice才能够实现应用程序之间的通信。对于应用端，使用Java Message Service（JMS），JMS API是一个建立在java EE上的信息标准，允许所有的应用对象创造，发送，接受和读取信息。它可以使分布式通信轻松地连接、十分可靠并且能实现异步。Open Message Queue（Open MQ）是JMS的一个标准实例和消息等待队列，JMS就是依赖它来运作的。

总结

总的来说，设计是把分析模型转换为设计模型的过程。在架构设计中，我们根据项目的设计目标和相关的设计原则，对系统进行了合理分解，形成了不同的系统层次和各类构件，并对其中的核心元素和架构机制进行了定义，为下一步的数据设计提供了基础。