**情感词汇分析系统**

**总体架构说明书**

|  |  |  |
| --- | --- | --- |
|  | | |
| 98k研究团队 | **文档名称** | {情感词汇分析系统}总体架构说明书 |
| **文档编号** |  |
| **版 本 号** | V1.0 |
| **密 级** | 机密 |
| **日 期** | 2018/6/28 |
|  | | |

**文档历史发放及记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 发布 | 史谨 | V1.0 | 20018-6-285 |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |

**版权说明**

《{情感词汇分析系统}总体架构说明书》的著作权、版权和知识产权属于天柏宽带网络科技有限公司所有，并受《中华人民共和国著作权法》、《中华人民共和国专利法》、《计算机软件保护条理》、《知识产权保护条例》和相关国际版权条约、法律、法规，以及其它知识产权法律和条约的保护。

《{情感词汇分析系统}总体架构说明书》包含的权利，包括所有权、著作权、版权、专利权、商业秘密、商标、知识产权和其它所有权属于天柏宽带网络科技有限公司所有。任何单位和个人未经天柏宽带网络科技有限公司授权不得翻印和出版本手册，否则将视为非法侵害，我公司保留依法追究其责任的权利。

98k研究团队

20018年6月

**目 录**

[1. 背景 4](#_Toc1189)

[2. 需求 4](#_Toc31635)

[3. 业务特点及开发约束 5](#_Toc23356)

[3.1. 业务特点 5](#_Toc27476)

[3.2. 开发约束 5](#_Toc30851)

[4. 术语与参考资料 5](#_Toc30849)

[4.1. 术语 5](#_Toc8729)

[4.2. 参考资料 6](#_Toc2061)

[5. 开发规范 6](#_Toc20395)

[5.1. 编码规范 6](#_Toc26449)

[5.2. 开发环境与开发工具 6](#_Toc1034)

[5.3. 开发目录及类名、包结构命名规范 7](#_Toc13651)

[5.3.1. 开发目录规范 7](#_Toc13265)

[5.3.2. 包结构命名规范 8](#_Toc18236)

[6. 双向业务架构总体设计 10](#_Toc20088)

[6.1. 业务实体设计 10](#_Toc27066)

[6.1.1. SSO 10](#_Toc25800)

[6.1.2. 接口服务 10](#_Toc11795)

[6.2. 系统性能设计 11](#_Toc1)

[6.2.1. 系统精确度 11](#_Toc10552)

[6.2.2. 系统扩展性 11](#_Toc5948)

[6.2.3. 系统移植性 12](#_Toc31779)

[6.2.4. 系统复用性 13](#_Toc15563)

[6.2.5. 系统维护性 13](#_Toc30765)

[6.2.6. 系统安全性 13](#_Toc11171)

[7. 技术框架 14](#_Toc1883)

[7.1. 技术框架图 14](#_Toc23832)

[7.2. 技术选型 14](#_Toc32674)

[7.2.1. J2EE技术路线 14](#_Toc20849)

[7.2.2. Dot net技术路线 18](#_Toc433)

[7.2.3. 其它因素 20](#_Toc18971)

[7.3. 使用人群及使用约束 20](#_Toc18126)

[7.4. 框架设计思想 20](#_Toc27275)

# 背景

当今社会发展与时俱进，社会上舆论信息各种各样，层出不穷，对那些舆论信息进行数据分析必不可少，我们这项产品采用机器学习的先进算法对所获取的各种各样的信息进行逐一的分析，将其进行分类归并，并且做出一些当事人想要做的一些基础操作，我们这个产品简单易操作，上手运用起来相当的简单，按照当前社会上的快节奏，我们这项系统分析数据速度快，简单方便的特点肯定会十分的受欢迎，会收到各大网站的欢迎，更多的是应用于一些新闻发布的网站，绝对是炙手可热的产品。

通过实施本项目可以为客户带来如下收益：

* 适应未来多业务的发展规划
* 加快业务上线周期
* 适应互动业务带来的商务模式变化，支持开放式的运营模式；
* 减少新业务上线对运营支撑系统的影响；
* 简化业务提供者、应用开发商工作，从而降低运营商成本
* 提升用户界面的自动化维护机制，降低维护压力
* 加强对产品生命周期管理，适应业务快速变化的环境
* 水平化的多业务统一管理、避免垂直化架构带来的信息孤岛

# 需求

见《情感词汇分析系统用户需求说明书》。

# 业务特点及开发约束

## 业务特点

1. 现代网络的发展，让研究人员可以通过大数据分析，研究用户的兴趣爱好；
2. 本系统提供更方便的方式，为有需求的用户解析数据；
3. 人工智能技术的发展需要大数据以及数据解析技术的支持；

## 开发约束

1. 系统有限的资源限制：

由于数据需求较多，部分网站禁止抓取数据；

研究数据保密性太多，导致数据解析不完全；

1. 各模块浏览器的限制：

不支持复杂的Javascript操作；

不支持CSS link；

# 术语与参考资料

## 术语

|  |  |  |
| --- | --- | --- |
| 关键词 | 全称 | 含义说明 |
| Json | JavaScript Object Notation | JSON(JavaScript Object Notation, JS 对象简谱) 是一种轻量级的数据交换格式。它基于 ECMAScript (欧洲计算机协会制定的js规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。 |

## 参考资料

|  |  |  |  |
| --- | --- | --- | --- |
| **文档** | **版本** | **作者/来源** | **备注** |
| 《情感词汇分析系统需求规格说明书》 | V1.0 | 98k研究团队 |  |
| 《情感词汇分析系统概要设计》 | V1.0 | 98k研究团队 |  |

# 开发规范

## 编码规范

见《J2EE编码规范》。

## 开发环境与开发工具

1. 操作系统
   1. Windows
2. 应用服务器
   1. Tomcat (8.5)
3. Java软件开发包
   1. JDK (9.0.4)
4. 开发工具版本
   1. Eclipse
5. 数据库
   1. Mysql
6. 版本控制工具
   1. TortoiseSVN (2.6.0.0)
7. 数据库建模及绘图工具
   1. Visio (v2013)

## 开发目录及类名、包结构命名规范

### 开发目录规范

1. src：存放源代码的目录，具体src中的包结构规范请参照5.3.2节的“类名、包结构命名规范”。
2. test：存放测试代码的目录，具体test中的包结构规范请参照5.3.2节的“类名、包结构命名规范”。
3. resources：存放资源文件的目录，具体为：
4. config：存放基本的配置信息。如：数据库连接信息，hibernate、mail、log4j等配置信息。
5. i18n：存放“国际化”资源文件。根据不同的国家语言（即不同的文件后缀）建立子目录。如：建立cn子目录存放与中文相关的资源文件。
6. spring：存放spring的配置文件信息。
7. 其它：如果使用的开源项目配置信息很少，可以放在config目录下；如果使用的开源项目配置信息很多，请按照与spring、struts相同的方式在resources目录下建立独立的子目录。
8. lib：存放工程所引用的jar文件。
9. webapp：存放与web工程相关的文件，具体为：
10. pages：存放通用页面。如：错误页面，标签、包引用片段页面等。
11. config：存放基本的配置信息。如：数据库连接信息，hibernate、mail、log4j等配置信息。
12. Resource\is：存放图片和css等样式文件。按照功能模块划分子目录，要与WEB-INF\pages中的子目录结构对应。
13. webapp\WEB-INF：存放安全性高，不可直接访问的web工程文件，具体为：
14. classes：存放打包编译后的文件。
15. pages：存放页面文件。按照功能模块划分子目录，要与..\js中的子目录结构对应。
16. 其它：在开发过程中，如果有与web工程相关的，且安全性高的可独立划分成一个子目录的分类，请在webapp下建立相应的子目录。
17. build.xml：打包发布用的xml文件。如目录结构有增减，请相应的调整build.xml文件的配置信息。

### 包结构命名规范

#### 包结构及包的命名规范

1. 数据层
2. edu.zut.cs.emotion.admin.\*\*\*
3. edu.zut.cs.emotion.dao.\*\*\*
4. edu.zut.cs.emotion.test.\*\*\*
5. 业务层
6. src/main/java

edu.zut.cs.emotion

* Admin.service.dao
* Admin.service.Impl

1. Src/test/java

Edu.zut.cs.emotion.admin.service

1. 展现层
2. edu.zut.cs.emotion.admin.web.spring.controller
3. Webapp

Folder

index

#### 相关约束：

1. 组件相关：edu.zut.cs.emotion.\*

“\*”为各种组件的包结构。如：dao、base、web ,service等。一般于程序框架开发人员有关。

1. jar包规范：
2. 针对个项目的工具资源放在每个项目下的jar包；

Edu.zut.cs.emotion.\*.jar（“\*”为各项目名称）

1. 所有项目均可共用的公共工具资源放在lib的jar包下；

Edu.zut.cs.emotion.jar

1. 各项目组的成员只有修改相关项目下jar包内工具资源的权限；
2. 各项目组的成员没有提升jar包内工具资源使用范围的权限；
3. 指定公共工具资源管理员来维护lib的jar工具包；
4. 其它：

开发人员可根据实际情况按照以上的规范与约束进行扩展。

#### 类的命名规范：

1. 实体定义：按照实际的意义为实体类命名，并放在数据模型包中（model）。如：User、Group等。
2. 实体操作：按照实体名+”Manager”方式命名，并放在数据操作包中（dao）。如：UserManager、GroupManager等。
3. 业务逻辑层的类放在services包中，接口采用以业务为中心的有意义的名字。如：Operation等。而实现类也在同一个包中，以业务接口名+”Impl”方式命名。如：OperationImpl等。
4. 在给service命名的时候，要在后面加上”service”后缀。

# 双向业务架构总体设计

## 业务实体设计

### SSO

#### 实体定义：

单点登录（Single Sign On），简称为 SSO，是目前企业级用户身份认证整合的解决方案之一。SSO的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。它包括可以将这次主要的登录映射到其他应用中用于同一个用户的登录的机制。

#### 设计思想：

SSO应用在两个方面：

1. 前台用户认证。用户首先通过SSO登录双向系统，然后通过PFS去访问各个业务。在此过程中，用户不需要再次向各业务中去登录。
2. 后台管理员认证。由于一些业务的管理不是在PMC，而是采用Link方式从PMC统一进入，这样在每次进入各业务后台管理页面时，也要认证用户身份。通过SSO的控制，可以实现单点登录。

### 接口服务

#### 实体定义：

如果配置管理配置了外挂系统的信息后，接口服务后台就可以直接读取外挂系统的配置信息，进而显示出它的相关接口。

#### 设计思想：

接口服务有自己独立的后台管理，脱离PMC仍然可以支持其它的应用。接口服务同PMC的配置管理同样可以对外挂系统信息进行配置，写入到数据库的同一个地方。在接口服务启动的时候先去读取这些信息，然后开放相应的接口服务。

其中包括两类接口：一个是获取内容的接口。例如：元数据信息，产品信息等。另一个是业务要素信息。

## 系统性能设计

### 系统精确度

|  |  |
| --- | --- |
| **性能指标** | **精确度说明** |
| 时间特性 | 响应时间：用户操作，最多3秒内得到响应。若页面很复杂，或逻辑很复杂，至少能先有部分结果，或者友好的提示。 |
| 性能特性 | 在页面响应时间2秒内，数据库有反应 |
| 规模特性 | 数据数量达到大数据要求 |
| 空间特性 | STB端是嵌入式系统，可用资源非常有限，在开发时特别要求软件必须简洁；  前端服务不受限制；  对服务器的硬件内在要求不超过4G； |

### 系统扩展性

#### 在数据持久层方面

采用开源的Hibernate 持久层组件，该组件是有名的OR-Mapping工具，使用该组件可以对性能做灵活的配置和性能优化，可以对一级缓存进行管理，可以开启二级缓存、控制事务策略，有灵活的1对1 关系、1对多关系的映射。

#### 在数据库设计方面

对于海量数据的存储采用分区的策略，可以是范围分区、列表分区、散列分区、组合范围散列分区等。

为经常查询的列建立索引以此来加快访问数据库表中特定信息的速度，避免或简化排序操作，避免相关子查询，使用存储过程加快数据库逻辑的牏速度。

针对特定的数据库做特殊优化。

#### 消息中间件方面

使用ActiveMQ消息中间件组件，使得消息的生产者与消息的消费者相解藕，保证数据在高并发时系统的可靠性，并采用P2P、Publish/Subscriber两种方式传递消息，保证数据的持久化及分发策略。

#### 在代码方面

遵循J2EE相关规范进行开发，使用Spring 等AOP框架，灵活的集成或扩展应用。

使用缓存组件，对经常被访问的数据进行缓存，加快查询及展示的速度。

### 系统移植性

本项目使用Java语言，采用Spring + Hibernate + Struts + Tomacat + ActiveMQ的开源框架，遵循J2EE框架开发。

由于Java的跨平台特性使得本项目所开发的系统可以运行在Window系列、Linux系列操作系统之上。

开源的Hibernate 持久层组件来支持跨数据库的操作，Hibernate 组件特有的 HQL语言在执行时被解释成特定的数据库SQL语言，保证了跨数据库的特性。

遵循J2EE规范进行开发，使得开发后的B/S架构Web 系统可以运行在任何遵循Sun公司（关于Web 应用服务器的）J2EE规范所开发实现的应用服务器之上，这些服务器可以是开源的也可以是商用的应用服务器，如：Apache Tomcat，BEA Weblogic，IBM WebSphere等。

使用遵循J2EE规范之JMS v1.1规范的ActiveMQ消息中间件组件，使得在未来可以根据需要使用任何支持JMS规范的中间件组件代替ActiveMQ组件，实现可移值性。

### 系统复用性

从设计的角度，纵向方面，系统采用组件化的方式的进行设计，合理的将公告、电视回看EPG、VOD EPG等业务组件化，保证开发后的系统可以“有选择性”的复用到未来相类似的Portal系统中。

横向方面，系统采用Persistence + Business (两层business，使用business delegate) + Presentation 多层设计，针对业务需求科学的决策，开发可复用的核心技术框架，这样对于新的应用可直接复用并基于核心技术框架进行开发，即使要对核心技术框架修改也不会对应用产生大的影响。Business层基于Persistence之上根据业务进行开发不需要关心采用何种展现层技术进行展现，使得设计出的系统灵活、可复用。展现层采用MVC模式的Struts组件，使得视图与业务很好的分离开，视图中的action提交给定义好的 action name，无需关心该action的具体实现，使得“功能代码”与“视图（界面）”很好的分离。

### 系统维护性

系统采用组件化的方式的进行设计，将业务组件（套件）化，采用Spring作为集成框架，针对具体需求可以灵活的装配业务组件，达到灵活、可配置。

系统提供PMC(Portal Center Manager)对系统的后台进行参数配置、管理，PCM提供配置管理，应用管理，管理员管理，操作日志，统计分析，减少后台管理员的维护工作量，让系统的配置、管理变得容易简单、可视化。

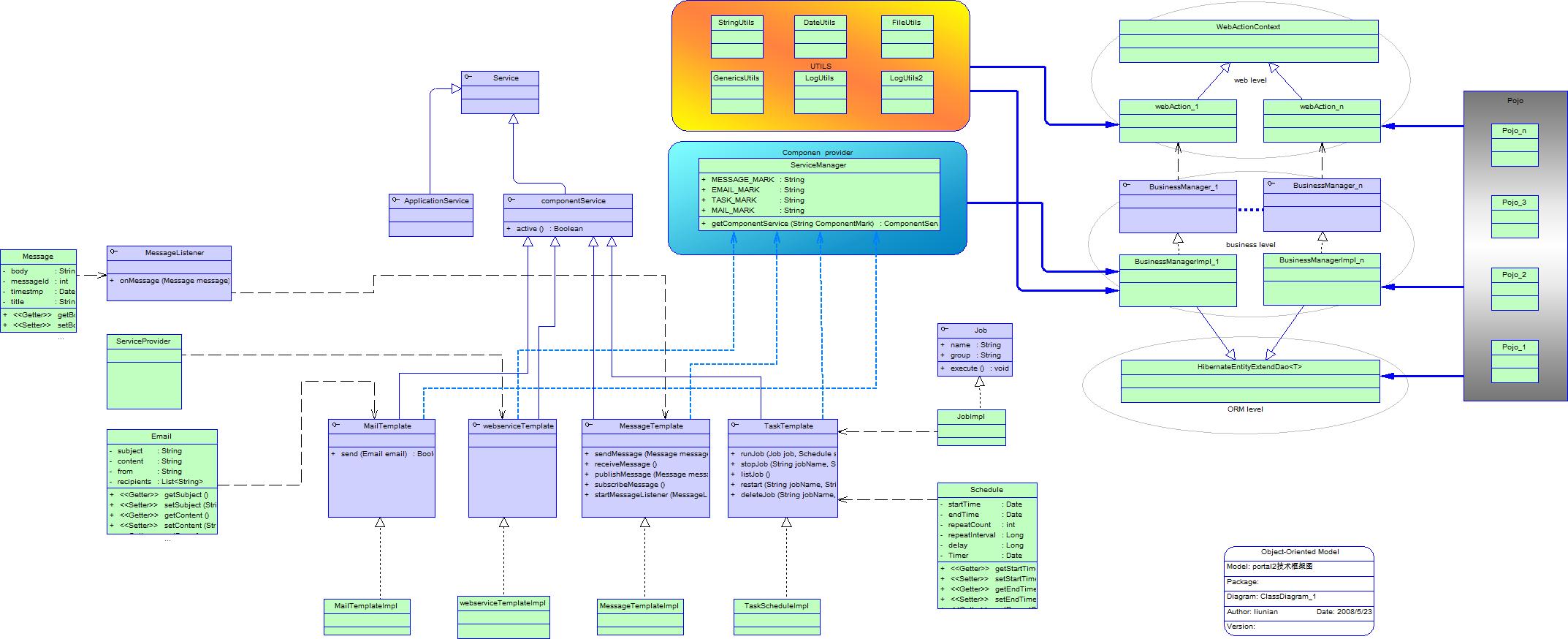
### 系统安全性

系统支持HTTPS协议，使用随机生成的token作为摘要，然后MD5算法对登录的STB机顶盒用户的密码和摘要进行加密，保证身份信息的传递安全。

采用SSO单点登录对所有的登录使用进行身份验证，身份验证的机制可以是数据库验证，也可以是LDAP验证。用户身份验证的全过程使用凭证进行操作，包括服务凭证（ST）、代理授权凭证（PGT）、代理凭证（PT），构成完整的凭证链，保证了系统的安全。

# 技术框架

## 技术框架图



## 技术选型

### J2EE技术路线

#### 开发语言特性

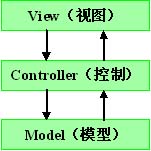
1. 技术来自于多家公司；
2. 开源产品众多，免费框架居多，硬件和中间件需付费；
3. 成果众多，相应的最佳实践设计模式层出不穷；
4. 平台移植性好，支持所有操作系统，这一方面成本降低；
5. 开源社区活跃，可以快速解决开发过程中遇到的问题。

#### 开发框架

1. MVC展现层开发框架

整个项目团队对于MVC框架较为熟悉的就是Struts，一方面Struts作为业务的通过选择，另一方面可以较少开发周期并直接带来成本的下降。

模型-视图-控制器（Model-View-Controller）是80年代出现的一种软件设计模式，现在已经被广泛的使用。它强制的把应用程序的输入、处理、输出分开，三个核心模块M-V-C分别负担不同的任务。



模型（Model），模型是应用程序的主体部分。模型表示业务数据和业务逻辑，一个模型可以为多个视图提供数据，提高了应用的可重用性。

视图（View），视图是应用程序中用户界面相关的部分，视图向用户显示数据，并能接收用户的输入数据，但它并不进行任何实际的业务处理。

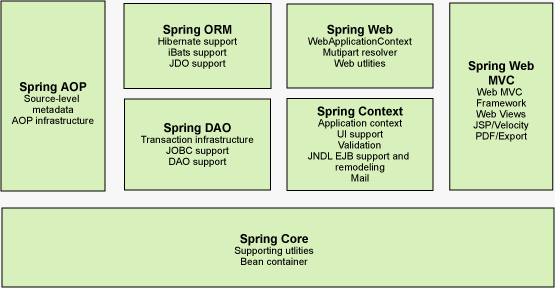
控制器（controller) ，控制器工作就是根据用户请求，调用相应的模型组件处理请求，然后调用相应的视图显示模型返回的数据。

选择这个框架的原因是程序员都很熟悉struts，而且网上相关的文档也很多，框架的的后续支持高，使用人群大，问题解决很容易。

1. Spring 集成框架

Spring 是一个开源框架，是为了解决企业应用程序开发复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许您选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架。

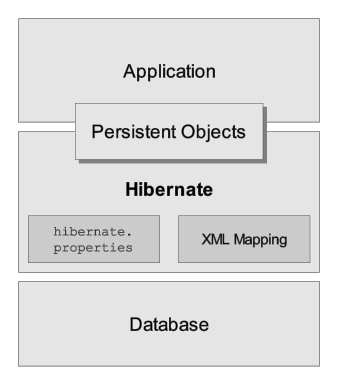
Spring作为AOP, IOC框架， 其采用分层设计可以方便的集成各组件。



使用Spring主要是为了发展自身的核心组件。Spring已经集成了众多优秀的开源组件，而且大部分开源框架都会集成Spring。使用人数众多，问题解决容易。

1. Hibernate 持久层框架

Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了轻量级的对象封装，使 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。它不仅提供了从 Java 类到数据表之间的映射，也提供了数据查询和恢复机制。



相对于使用 JDBC 和 SQL 来手工操作数据库，Hibernate 可以大大减少操作数据库的工作量。 另外 Hibernate 可以利用代理模式来简化载入类的过程，这将大大减少利用 Hibernate QL 从数据库提取数据的代码的编写量，从而节约开发时间和开发成本 Hibernate 可以和多种Web 服务器或者应用服务器良好集成。

#### 数据库

由于Mysql对于海量数据的支持能力不足，所以在这里优先考虑选用数据库Hibernate

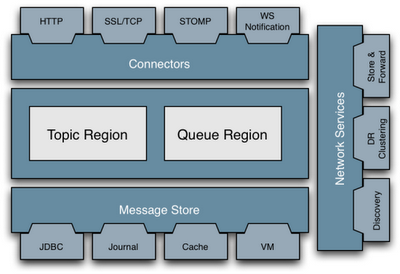
#### 操作系统

在企业级的应用中只能选择Window Server 系列产品，如：Window Server 2008，Window Server 10等，但Window系列操作系统的安全性较Unix，Linux系列操作系统低一些。

#### JMS消息中间件

数据传输时，需要考虑数据在高并发时系统的可靠性。通过队列机制来保证数据的持久化及分发策略。

Apache ActiveMQ是遵循Sun JMS v1.1规范开发的开源消息中间件产品，一个快速的开源消息组件，支持集群，同等网络，自动检测，TCP，SSL，广播，持久化，XA，和J2EE1.4容器无缝结合，并且支持轻量级容器和大多数跨语言客户端上的Java虚拟机。ActiveMQ的发布得到了Apache2.0的许可。



ActiveMq在以往项目中有过良好的表现、并且是一个维护良好的开源项目，所以优先选用ActiveMq。

#### 应用服务器

可以采用开源的Tomcat作为开发和部署应用服务器，如果用户对Web 应用服务器要特殊的要求，可以按照客户的需要进行选择。Tomcat是遵循Sun关于应用服务器的相关规范开发的开源软件，开源社区比较活跃，在使用的过程中如果有任何问题都可以很容易的找到解决方案。

从部署架构上讲，应用服务器可以与Web服务器相连，共同为客户端服务，或应用服务器独立为客户端服务。

### Dot net技术路线

#### 开发语言特性

1. 技术来自于一家公司；
2. 软硬件均需要付费；
3. 多数设计模式最佳实践灵感来源于J2EE阵营；
4. 仅支持Windows系列操作系统；
5. 无开源社区支持，是以框架开发者为主导的设计；
6. 门槛很低，使用方便，学习成本较低；
7. 新技术更新较慢。

#### 开发框架

数据持久层Nhibernate和IbatisNet这两个都是非常优秀的数据持久层，Nhibernate是优秀的Hibernate的dotNet移植版本，在开源社区具有非常高的人气，IbatisNet是Data Mapper框架，在dotnet的开源社区一样是非常受欢迎的一个工具。Nhibernate用于支持非常好的面向对象的设计的模型，IbatisNet用于支持应用程序的移植，这样就具有更大的弹性。

IOC容器Spring.Net和Castle，这是两个dotnet非常优秀的IOC容器。Spring.Net同样是Java的Spring 的移植版本，目前的版本是0.6。

Castle则是dotnet下出现新的IOC容器,它的功能，成熟度方面比Spring.Net好得多，框架中准备采用Castle.最后的平台的技术架构就是Nhibernate/IbatisNet + Castle + ASP.NET。

#### 数据库

与Window系列产品结合最优的数据库选择是 SQL Server数据库系统，可以充分利用该数据库许多特性和让数据库的优势发挥到最大化。

#### 操作系统

在企业级的应用中只能选择Window Server 系列产品，如：Window Server 2003，Window Server 2008等，但Window系列操作系统的安全性较Unix，Linux系列操作系统低一些。

#### 消息中间件

Microsoft BizTalk Server 用来连接内部和跨组织的系统，包括交换数据和梳理跨多系统的业务流程，集成商业规则与人力工作流服务。Microsoft BizTalk Server的实时、端到端的供应链管理涉及企业内部或跨企业的每个客户，系统，人和流程，使用户能够根据来自于地理上分布的集成系统的实时数据产生可靠的商务决策。

#### Web 服务器

只能使用Microsoft 公司的IIS(Internet Information Server)。

### 其它因素

1. 本项目复用“Portal二期项目”的工作成果，“Portal二期项目”采用J2EE技术路线；
2. 项目组成员对J2EE路线相关技术较熟悉；
3. 项目组外其他技术人员也都对J2EE路线较熟悉，这样，在人力资源紧缺的时候可以很快的得到补充；
4. 客户指定的硬件设备不适于安装和使用Windows系列操作系统，也就不能很好的支持.NET 语言；
5. Dot net 技术路线所需的软件产品通常依赖于商业的操作系统、数据库、消息中间件、开发框架等软件产品。

## 使用人群及使用约束

1、框架开发人员

框架开发人员可以对整个框架进行修改。

2、程序开发人员

程序开发人员只能利用暴露的接口。

## 框架设计思想

在整体框架设计上采用spring的IOC思想，使得各层解耦。通过动态注入，可以随时切换底层的实现技术。但程序开发人员根本无法感觉到差异的存在。框架设计的原则是：简单、实用、透明、规范。

1. ServiceManager

ServiceManager为静态类。通过ServiceManager可以直接获得各种组件。此静态类事直接暴露给程序开发人员使用的。

1. componentService

组件服务层是通过ServiceManager获得的。其中包括：MailTemplate、webserviceTemplate、MessageTemplate、TaskTemplate。此层组件是可以互相独立的，并且直接提供给业务逻辑层使用的，不与其它层有交互的。这四个接口是直接提供给程序开发人员使用的，并且为了每个接口提供了配套使用的DTO类或接口。用户不知道底层实现的技术和方式。

1. Utils

公用组件。是提供给每一个层次使用的组件，无需ServiceManager管理。这一层包括transform、log、configuration等组件，可直接提供给各层开发人员。

1. DAO层

DAO层是直接服务于业务逻辑层的供开发人员直接使用的。由于springside已经对hibernate做了三层封装，并且在对数据库的操作上是具备一定的复杂性和不可预见性的，所以我们不打算在此基础上进行进一步的封装，而是直接提供给程序开发人员使用。但是，由于springside对hibernate的封装有三层方式，使得开发方式有多种选择。为了使得程序员在开发的时候能够统一，我们提出了相关约束来规范：

1. 抽象出每个模块所使用的数据对象（POJO）
2. 采用POJO与POJO manager一一对应的原则
3. 采用POJO manager继承HibernateEntityDao<T>的方式（此类已经由springside封装）
4. 如果有功能扩展，在每个manager上进行