hihsoft 手拉手平台篇

个性化解决方案的核心

------业务基础平台

文件状态： 【】 草稿 【 】 正式发布 【√】 正在修改中	文件标识	hihsoft-flat-000001		
	文档名称	业务基础平台总体设计方案		
	版本	日期	作者	备注说明
	1.0	2013-6-6	hihsoft	
	密级	**公开**		
	应用范围	喜欢hihsoft业务基础平台的拉丝们		
	版权所有	JavaHih为广大技术人员提供软件组件化架构和设计的交流平台，集思广益，打造良好的组件化产品，并有专业人士对文档进行校审，文档以LGPL等开放协议发布，允许并鼓励在网上转载及发布，并寻求与更多技术社区和网站的合作推广，该文档的所有权属于JavaHih手拉手研发团队。		

目 录

1 业务基础平台介绍

1.1 前言

今天你百度了吗？google 了吗？搜索无处不在、它与我们同在。互联网飞速发展，面对浩瀚的知识海洋，利用搜索神器来获取知识，已经成为我们生活的一部分，然而面对一堆的搜索结果，手气不错问题解决；手气差在搜索结果中耗时耗力，却问题没解决。作者也是千万程序猿中的一员，常为此事苦恼，坚信实践出真知、坚信只有从实际的项目案例的角度去剖析技术才是解决技术问题的必杀器。才是学习新技术的捷径。开源技术无处不在，开源与我们同在，hihsoft 业务基础平台是一个站在开源巨人的肩膀上，引导程序猿们使用开源技术的一个实战型平台。

Java 手拉手团队成员都是从事软件行业 10 余年，在电信、银行、电子政务、智能交通、物流、电力等不同行业积累了宝贵经验;在软件分析、架构设计、开发和管理经验上都有着自己独到的见解。领导完成多个百万级大型项目的设计和开发。熟练使用 Oracle、DB2、MySQL 等数据库，熟悉数据仓库和数据挖掘技术。Java 团队从实际的项目中提纯技术精要，为程序猿助力指引、分享经验、共同进步。将编写一系列面向开发人员技术手册帮助读者从菜鸟升级到大牛，钱途、钱途双途发展、打出码农的天下。

“面向开发人员，讲解对开发人员最有用的知识、拒绝废话连篇”是本网站的宗旨。取之于开源、用之于开源，教会你我他怎样使用优秀开源软件是本站的宗旨和服务理念。

1.2 平台总体设计原则

架构是一种权衡。基于产品组件化的业务基础平台拟结合具体的实施项目，从行业角度、技术角度综合考虑，力求在软件的伸缩性、可维护性、可扩展性、易测性，可靠性、容量、安全及性能要求上、团队成员技术整体素质、开发效率上等诸多因素间找到一种最佳平衡点，最终达到能良好的解决用户需求（包括功能性和非功能性）之目的。

- 先进性原则：采用国际先进的技术路线和体系结构。
- 实用性原则：能满足综合平台功能需求的同时，系统设计过程还应考虑到易于操作、易于使用、界面友好等问题。
- 标准化原则：在系统的开发过程中必须采用标准化的原则，遵循国际标准、国家标准、行业标准等。
- 开放性原则：采用成熟的第三方开源软件平台和工具进行开发，并预留必要的对外系统接口。
- 兼容性原则：保证系统能跨平台运行，可以运行在各种操作系统平台上，如 UNIX、Linux、Windows NT 等平台，并支持兼容多种浏览器（IE6+、Firefox、Chrome 等）。
- 共享性原则：提供必要的系统接口，数据资源和服务可与其它系统共享。
- 安全性原则：要求按有关规定采用物理隔离网闸、防火墙、防病毒和防黑客软件

保障系统安全，防止 DOS 攻击，并在软件设计上保证系统和数据的安全。

- 保密性原则：对数据库敏感字段内容加密；确保未经许可不可获取数据库信息。
- 可靠性原则：数据定期增量备份等技术手段确保系统稳定运行。
- 可扩展性原则：系统的设计和建设应充分考虑网络、硬件的扩展及系统二次开发的需要，并支持未来可能出现的新业务的需要。
- 可维护性原则：有详细的调试跟踪信息记录，便于问题跟踪和系统调试

1.2.1 KISS 原则

本业务组件基础平台，本着在满足功能及各个软件硬性指标的前提下，尽量将系统框架设计得简单、稳定、安全。即万丈高楼平地起，根基至关重要。

- Keep It Simple，Stupid，简单就是美。
- 平台基准：制定统一的 UI 设计标准，遵循清晰易用的原则，合理布局，界面结构清晰,坚持平淡中显专业。支持多种系统整体风格(themes)的选择。
- 界面尽可能人性化，增强系统易用性，注重用户体验性。
- 前端采用通用的 HTML 语法，层(Div)和表格(Table)相结合的简单前端开发，吸收国内外做得比较好的前端 UI 组件例如 JQuery miniui 和 Jquery easyui、JUI 富客户端框架的架构和设计。

1.2.2 强内聚，松耦合

主要指业务基础平台架构或代码的组织形式：

- 强内聚：一个层、业务组件、或具体的类、方法，应该只完成一项任务而且圆满

完成

- 松耦合： 基于松耦合的组件化开发，不同的业务组件之间通过标准的接口调用，具体组件分层设计、业务模块、或具体的类、方法原则上都是强内聚、松耦合的

1.2.3 面向组件

- 面向构件设计，形成软件公司或企业内部组件库：分为通用（公共）组件和业务组件、服务组件，在不同项目中共享应用
- 研发出以 hih 开头的通用构件，hih 表单组件、安全组件、权限组件、图表组件等
- 注意组件库的持续性，不断提升和优化构件库

1.2.4 粒度适当

在系统框架里面，有各种层次的粒度：

- **代码粒度：**从性能角度、维护角度考虑类的大小、包的大小、js 脚本大小等
- **数据粒度：**从数据库中取得的数据量的大小、传输调用过程中数据量的大小
- **进程粒度：**业务进程的大小长短
- **事务粒度：**事务涉及的跨度、持续的时间
- **版本粒度：**向版本库提交代码的频率大小
- **安全粒度：**意识系统的安全性的重要性，把握好安全的粒度，从系统在安全性、维护性、性能上找到平衡
- **业务组件粒度**太小，造成组件数量多，组件之间交互多，管理困难，性能低下；业务组件粒度粗，功能复杂，功能之间关系紧密，升级困难（可以独立升级往往

会作为确定一个业务组件范围的重要因素），很难实现重用。因此找到一个合适的业务组件粒度至关重要。

在系统设计的时候，要把握好各种粒度的大小。当粒度太大时，维护困难，耦合也可能过于紧密，可以从设计上加以考虑划分成更合适的粒度。粒度也不要太小，以免增加设计和代码实现的复杂性。

1.2.5 面向对象的设计原则

面向对象程序设计中的金科玉律（五原则）

- **单一职责原则：**其核心思想为：一个类，最好只做一件事，只有一个引起它的变化。单一职责原则可以看做是低耦合、高内聚在面向对象原则上的引申，将职责定义为引起变化的原因，以提高内聚性来减少引起变化的原因。职责过多，可能引起它变化的原因就越多，这将导致职责依赖，相互之间就产生影响，从而大大损伤其内聚性和耦合度。通常意义下的单一职责，就是指只有一种单一功能，不要为类实现过多的功能点，以保证实体只有一个引起它变化的原因。
- **依赖倒置原则：**其核心思想是：依赖于抽象。具体而言就是高层模块不依赖于底层模块，二者都同依赖于抽象；抽象不依赖于具体，具体依赖于抽象。
- **开放 - 封闭法则（OCP）可扩展**，即"对扩展是开放的"（Open For Extension）- 模块的行为可以被扩展，以满足新的需求。不可更改，即"对更改是封闭的"（Closed for Modification）- 模块的源代码是不允许进行改动的。
- **接口隔离原则：**其核心思想是：使用多个小的专门的接口，而不要使用一个大的总接口。

- **Liskov 替换原则：**其核心思想是：子类必须能够替换其基类。这一思想体现为对继承机制的约束规范，只有子类能够替换基类时，才能保证系统在运行期内识别子类，这是保证继承复用的基础。在父类和子类的具体行为中，必须严格把握继承层次中的关系和特征，将基类替换为子类，程序的行为不会发生任何变化。同时，这一约束反过来则是不成立的，子类可以替换基类，但是基类不一定能替换子类。

1.2.6 面向接口

- 针对接口编程，而非实现：接口是一种契约，但却可使派生类灵活的实现
- 方便各层次的开发人员、整体协调，提高开发效率
- 各层之间的通信使用接口，有利于代码的解耦

1.2.7 面向服务

软件即服务 SAAS

面向服务的体系结构(Service-Oriented Architecture, SOA)

- SOA 实现降低各子系统之间的耦合
- 方便拓展业务，供外界系统调用

1.2.8 面向方面编程

面向方面编程的体系结构(AOP)，充分利用 DI、AOP 思想，使框架更灵活。

- 灵活运用切面的概念，专注解决框架统一性问题：日志、事务、安全等
- AOP 在架构系统时，更容易适合 B/S 的多层开发模式

1.2.9 巧妙运用设计模式

- Factory 工厂模式

- Builder 模式
- Single 单例模式
- ORM 对象关系映射模式

……

1.3 平台定位与目标

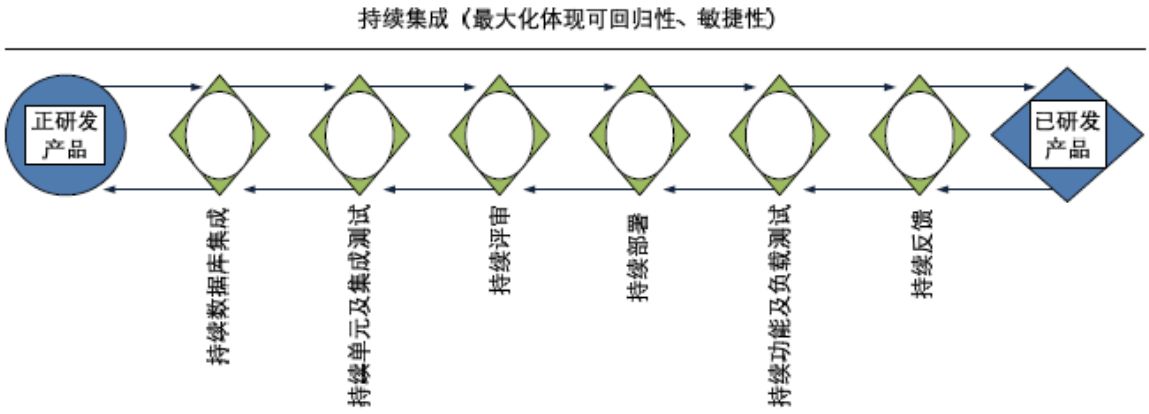
平台的目标有两个：一是通过提供丰富的可复用组件和图形化开发方式，降低应用开发人员对技术细节的依赖，在提高项目的开发效率的同时降低项目开发成本；二是通过业务基础平台规范和统一企业的技术开发框架，保障应用的高稳定性和高扩展性。最终打造一个轻量级、性能良好、快速开发的业务基础平台。

- 符合面向对象原则以及符合 J2EE 规范，满足开发环境和运行环境，适合多种角色的项目团队分层协作快速开发
- 利用提供模板语言制作代码机，生成核心代码，实现快速研发
- 持续改进手拉手 hih 组件库、在不同的项目中推广应用，提高项目实施的整体质量

组件式平台的构想如下图所示：

1.4 端到端研发模型

架构基于持续集成的全生命周期研发模型



完整的研发过程（静态）比如：其内容涵盖软件研发的全生命周期，即面向“设计、开发、测试、研发流程、运行时监控、问题诊断”等环节

持续集成的实施（动态）。比如：利用 Hudson、CruiseControl CI 服务器动态

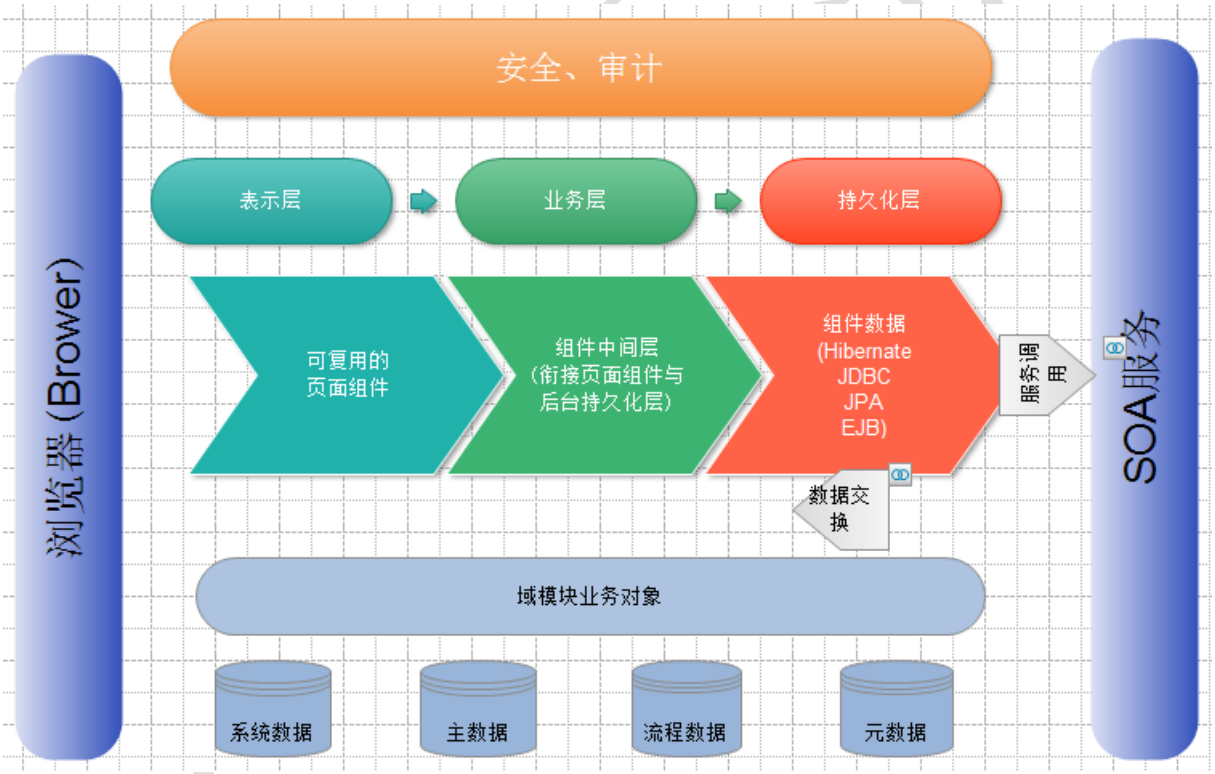
管理起整个研发工作的各个环节

1.5 组件平台实施准则

标准化->流程化->系统化->智能化（BI）

1.6 组件基础平台总体技术

本着不重复造轮子的思想，选用优秀的开源框架，教会你、我、他怎样使用优秀的开源框架，充分发挥开源的效能。



说明：系统采用 B/S 多层技术架构，分为表示层、业务逻辑处理层、数据访问层（持久化层）、数据存储层、接口层。

- **表示层:**由各个用户界面组成，具有交互的功能，负责接收用户的输入和显示系统的输出。采用 AJAX 技术实现局部刷新，避免频繁的整个页面重新装载，提供系统响应速度并改

善用户体验。网页设计时遵循 W3C 标准，以实现对主流 Web 浏览器的兼容支持。

- **业务逻辑处理层:**实现具体的业务处理逻辑，以及安全控制和权限检查等。采用开源 Spring 框架，为应用程序开发提供一个集成的、分层的框架，实现灵活的事务、会话和应用程序配置管理。
- **持久化层:**使用 Hibernate 封装对业务数据库的存取访问、复杂查询通过 JDBC 来实现。数据访问层:实现对数据存储层的数据的存取访问，并通过系统接口，实现对外部系统的集成和数据访问。
- **接口层：**面向 SOA 服务编程，实现不同系统服务的大集成
- **合理利用缓存技术组件、**提高系统响应速度

1.6.1 Java 层 API 技术

采用 JDK1.6 以上版本

1.6.2 WEB 层技术

采用成熟的、灵活的 Spring MVC 框架设计（零配置）

均衡后期的维护及升级，页面应按区域来划分，实现页面组件化，主要采用 JQuery 来实现、定制自己的组件库。

加强数据校验功能：业务数据校验、表单数据校验良好的区分，表单数据采用 JS、JQuery 和 Bean Validator 相结合的原则、后台业务数据采用通用的模板页来显示。

页面缓存功能：采用 Oscache 技术来实现

1.6.3 持久化层技术

持久化层采用开源 ORM 框架，选用 Hibernate 技术,减少对数据库选用的依赖性

较为复杂的查询及系统报表制作可以采用封装 JDBC 框架下的公共类来处理

大数据量的查询，可以灵活配置缓存，提高系统响应速度，初选 EHCACHE

借助应用服务器的集群功能，实现负载均衡之目的

1.6.4 业务层技术

采用开源 Spring 框架，灵活运用 AOP、DI 思想。实现事务、缓存、日志（AspectJ）灵活配置

业务基础平台能够支持多数据源访问

1.6.5 子系统间调用技术

综合各方面考虑，采用 hessian RPC 远程过程调用框架与 Spring 整合

内部子系统可以使用数据库层面的交互来实现

开放业务接口时，采用 Axis 或 Cxf 开源 Web Serivce 来实现

1.6.6 数据库层技术

数据库层强化数据库的设计要求，严格按照数据库设计的规则来执行。

设计时应从结构、性能、移植性方面综合考虑

注重数据库优化机制

强化数据库安全

1.6.7 消息中间件

ActiveMQ 是 Apache 出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持 JMS1.1 和 J2EE 1.4 规范的 JMS Provider 实现，有如下特性：

多种语言和协议编写客户端。语言: Java,C,C++,C#,Ruby,Perl,Python,PHP。应用协议：OpenWire,Stomp REST,WS Notification,XMPP,AMQP

完全支持 JMS1.1 和 J2EE 1.4 规范 （持久化，XA 消息，事务)

对 Spring 的支持，ActiveMQ 可以很容易内嵌到使用 Spring 的系统里面去，而且也支持 Spring2.0 的特性

通过了常见 J2EE 服务器（如 Geronimo,JBoss 4,GlassFish,WebLogic)的测试，其中通过 JCA 1.5 resource adaptors 的配置，可以让 ActiveMQ 可以自动的部署到任何兼容 J2EE 1.4 商业服务器上

支持多种传送协议：in-VM,TCP,SSL,NIO,UDP,JGroups,JXTA

支持通过 JDBC 和 journal 提供高速的消息持久化

从设计上保证了高性能的集群，客户端-服务器，点对点

支持 Ajax

支持与 Axis 的整合

可以很容易得调用内嵌 JMS provider，进行测试

1.6.8 数据交换层

ETL，Extraction-Transformation-Loading 的缩写，中文名称为数据抽取、转换和加载。

大多数据仓库的数据架构可以概括为：

数据源-->ODS(操作型数据存储)-->DW-->DM(data mart)

ETL 贯穿其各个环节。

对数据的采集、转换、加载实行流程化、智能化管理

1.6.9 流程引擎

从商业和开源间综合各方面考虑，采用商业工作流，扩展性差，功能受限、但有技术支持；开源工作流有源代码、扩展性好，容易定制自己的工作流、没技术支持，投入研发的时间周期长。

技术调研后，采用基于 BPMN 2 业务流程建模的 Activiti 开源工作流，进行二次开发、形成流程引擎的核心部分

开发一套自定义工作流的 WEB 流程设计器

开发一套自定义表单工具,完成流程中相关的表单制作

1.6.10 搜索引擎

采用开源的 lucene

1.6.11 规则引擎

采用 JBOSS 规则引擎

1.6.12 计算引擎

灵活运用 AOP 切面，实现一个面向字符串的计算引擎，解决整个平台中的灵活性问题

1.6.13 消息引擎

实现类似 QQ 的即时通讯软件，解决不同行业的公司或企业间的消息畅通，保证数据传递的安全性。

① 即时沟通交流：方便、快捷地即时消息发送与接收，提供不同颜色字体的文字，提供个性化展示。

② 状态展示：提供查看联系人在线状态信息，可以方便、清晰地了解联系人的在线状态。

③ 组织架构：可清晰看到由树型目录表达的多层次企业组织架构，是实时更新的电子通讯录。

④ 联系人分组：支持常用联系人分组，把最频繁的联系人划入同一分组中管理。

⑤ 通讯录：提供公司外的联系人资料管理，可以进行分组，发短信，拨打电话。

⑥ 快速搜索栏：提供快捷搜索条，可以悬浮到桌面任何地方，提供帐号、拼音、中文姓名的模糊查找。

⑦ 消息通知：提供广播消息和系统消息，通知用户关键信息。

⑧ 历史消息查看器：对所有消息的历史记录进行查看、查找、归类。

。

1.7 定义、缩略语

1.7.1 术语定义

SN	简写	描述/全称
1	SSO	单点登录系统
2	hihsoft	Hand in hand software:手拉手产品组件
3	SAAS	Software-as-a-service：软件即服务

1.7.2 缩略语

1.7.3 参考资料

- 《计算机软件产品开发文件编制指南（GB8567-88）》
- 《JHIH 手拉手项目文档规范》

2 业务基础平台组件需求

2.1 组件需求

组件平台主要分公共组件和业务组件：

- 公共组件的公共形态包括最基础的数据传递的封装、基本的权限、组织机构、岗位、角色、组的划分、数据字典、元数据管理等

业务组件的公共形态包括以下内容：可以独立运行的系统或模块，业务组件的目的是以方便业务组件独立升级和减少不必要的组件之间的交互为基本原则，通过一定程度的分离，实现重用领域工件

人机交互的界面:界面组件（标签化）

业务组件提供的服务（面向接口来实现的服务）

具有自身的主数据元服务：监控功能、调试功能、异常处理功能

自身的数据库表结构及视图

数据交换支持提供服务接口和数据库访问

在 J2EE 环境中，可以实现业务组件全部由 Jar 包组成或基于 OSGI 的模式来实现业务组件的集成

以此为标准，将业务基础平台划分如下公共组件：

HIH Platform：用于业务系统快速开发

HIH Workflow：用于流程管理和监控

HIH Report：用于业务数据统计与分析、数据钻取

HIH Form：用于电子表单处理

HIH DataExchange（ETL）：用于数据交换

HIH Rule：用于企业中规则资产的统一管理和复用

HIH Search:用于内容管理系统的全文搜索

- 业务组件：

HIH ITIL：用于支撑运维管理、实现智能化运维

2.2 HIH Platform (hihsoft-sso)

所谓公共组件：每个项目执行时都需要用到的组件功能，可以根据项目要求以配置的方式加入，使每个公共组件考虑到公共服务组件的独立性，特别是保证每一个组件独立升级之后不会影响到其他的公共服务组件以及业务组件，因此需要对公共服务组件进行隔离。HIH Platform 属于底层构件，提纯最基础的公共组件，以通用性为设计宗旨。

将 HIH PlatForm(hihsoft-sso:业务基础平台)组件划分为如下几大区域来管理

安全管理区、元数据管理区、平台监控区、模块管理区、组件案例展示区（协助开发人员迅速掌握组件的使用）

2.2.1 我的桌面

个性化用户的主界面

利用 Portal 技术动态展现

2.2.2 安全管理区

- **用户管理**

用户基本信息、用户扩展信息（动态）。为兼容不同项目，不同的用户元素，支持用户自定义属性。实现思路可以参照 Oracle 的内置用户（Sys、System）

封装通用的用户查找 UI 组件供其他子系统调用:可以根据不同维度来查找维度下的用户集

按岗位查找、组织机构查找、在线用户查找、角色查找、用户状态查找

账号管理：密码重置、账号激活、锁定、是否过期、在线账号统计、会话管理

- **机构管理**

把组织机构和部门全部纳入机构管理，以目录结构树呈现，方便授予数据权限

封装通用 UI 机构树：多选、单选的机构树,供其他系统调用，避免重复劳动。

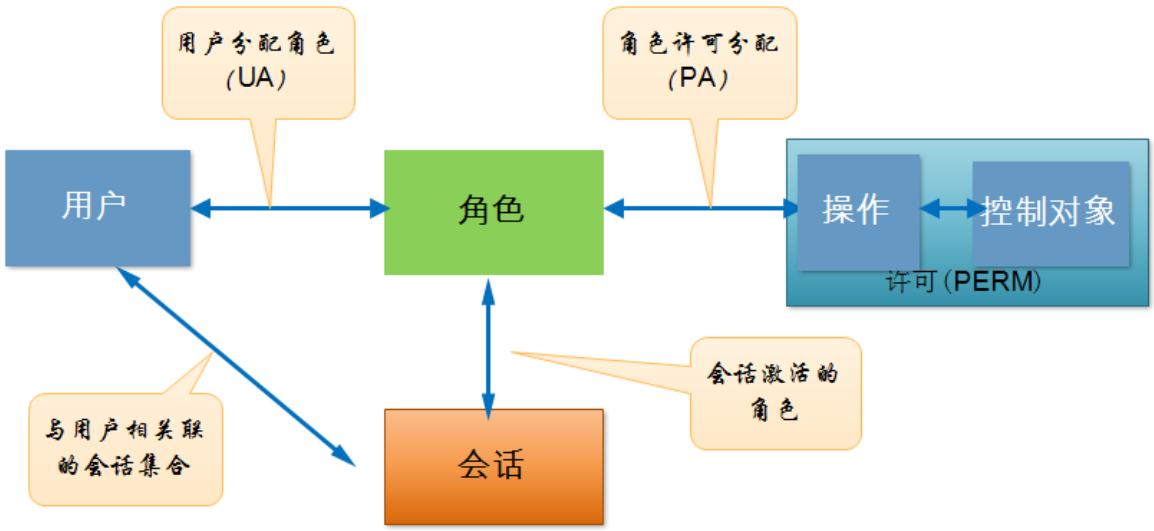
- **岗位管理**

自定义岗位，以目录树结构展示，岗位分配人员

- **角色管理**

内置角色(SYSDBA)和自定义角色，注意内置角色 SYSDBA 不允许删除，是否启用

- **权限管理**



RBAC 权限模型

RBAC0 定义了能构成一个 RBAC 控制系统的最小的元素集合。

RBAC1(引入角色继承关系)、RBAC2（引入责任分离关系）、RBAC3（角色继承+责任分离）都是在 RBAC0 上的扩展。

RBAC(Role-Based Access Control)权限模型：基于角色的权限控制,注重把握权限的粒度，粗粒度权限设计更具有重用性，控制系统模块及模块操作（资源分配）；细粒度权限一般涉及到具体的业务逻辑权限例如：页面元素权限控制、数

据集 DataSet 的数据权限控制等，此类权限与业务逻辑紧密相连，在实现粗粒度权限设计的同时，平台提供细粒度业务权限组件。在实际项目中要求支持如下的权限关系：

用户角色关系支持：同一个用户可以分配多个角色、同一角色可以分配给多个用户、支持双向授权；

支持分级授权：集团型（大集中）的项目，系统管理员多样化：存在省市区县分级管理员、同样存在集团、股份和分公司分级管理员。实现分级授权，分级管理。市级管理员在省级管理员授权的范围内实施市级系统授权，区县管理员在市级管理授权的范围内实施区县系统授权；

2.2.3 元数据管理区

元数据管理、数据质量监控等功能，主要实现技术元数据和业务元数据的管理。

元数据管理据具体需求来设定：据需而定

资源管理区

- **模块管理**

定义系统的模块（菜单）：设置模块的显示的图标、顺序

- **模块操作管理**

模块下的业务操作。以目录树的形式来展现：业务操作的图标，顺序等

- **附件管理**

通用的附件上传组件，例如：上传平台的相关帮助文档

2.2.4 平台监控区

- **登录日志**

监控系统上线用户，统计在线用户:包括用户的 IP 来源、用户名称、所属机构、最近登录时间、最近登出时间、是否正常退出等

- **审计日志**

对具体的业务，需要进行审计的功能加入审计日志。控制到元素级，提高业务的安全性。

- **业务操作日志**

用户最近执行的业务操作，在系统中留下历史记录

- **异常日志**

系统运行过程中，发生异常时，记录异常日志，方便解决异常，提高软件质量。

- **性能日志**

粒度可以细化到具体的方法花费的时间、优化系统性能。

2.2.5 组件案例展示区

演示各组件的使用

2.3 HIH Workflow

2.4 HIH Report

2.5 HIH Form

2.6 HIH Rule

2.7 HIH Search

2.8 HIH DataExchange（ETL）

ETL：数据交换系统

2.9 HIH ITIL

ITIL 运维管理系统