infohold-logo.png

智控-平台-规范

开发规范

密 级：□ 普通 □ 秘密 □ 机密 ■ 绝密

文档编号：IH\_CP\_SPC\_001

| 版本号 | 版本说明/变更理由/变更内容 | 作者/日期 | 审批人/日期 | 备注 |
| --- | --- | --- | --- | --- |
| 0.5 | 初始创建/C | 杨明/20110607 |  |  |
| 0.6 | 修改，梳理目录结构，更新技术规范/M | 张玉双/20160508 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

变更说明：C：Create，初始创建；A：Add，增加内容；M：Mod，修改；D：Del，删除

目录

[1 概述 4](#_Toc450469187)

[1.1 简介 4](#_Toc450469188)

[1.2 目的 4](#_Toc450469189)

[1.3 适用范围 4](#_Toc450469190)

[1.4 文档读者 4](#_Toc450469191)

[1.5 词汇表 4](#_Toc450469192)

[1.6 参考资料 4](#_Toc450469193)

[2 环境规范 4](#_Toc450469194)

[2.1 硬件环境 4](#_Toc450469195)

[2.2 操作系统 5](#_Toc450469196)

[2.3 开发工具 5](#_Toc450469197)

[3 技术规范 5](#_Toc450469198)

[3.1 系统架构 5](#_Toc450469199)

[3.2 技术规范 5](#_Toc450469200)

[3.2.1 SOA 5](#_Toc450469201)

[3.2.2 Micro Servcie 6](#_Toc450469202)

[3.2.3 Spring 7](#_Toc450469203)

[3.2.4 Spring Boot 8](#_Toc450469204)

[3.2.5 ReactJs 9](#_Toc450469205)

[3.2.6 React Native 9](#_Toc450469206)

[4 设计规范 9](#_Toc450469207)

[4.1 需求规格说明书规范 10](#_Toc450469208)

[4.2 系统原型设计规范 10](#_Toc450469209)

[4.3 数据库设计规范 10](#_Toc450469210)

[4.4 系统概要设计规范 10](#_Toc450469211)

[4.5 系统详细设计规范 10](#_Toc450469212)

[5 编码规范 10](#_Toc450469213)

[5.1 服务端编码规范 10](#_Toc450469214)

[5.1.1 Java编码规范 10](#_Toc450469215)

[5.1.2 Nodejs编码规范 10](#_Toc450469216)

[5.2 前端编码规范 10](#_Toc450469217)

[5.2.1 WEB前端编码规范 10](#_Toc450469218)

[5.2.2 App客户端编码规范 10](#_Toc450469219)

[6 开发过程规范 10](#_Toc450469220)

[6.1 开发任务分配、接收 10](#_Toc450469221)

[6.2 代码版本控制规范 11](#_Toc450469222)

[7 测试过程规范 11](#_Toc450469223)

# 概述

## 简介

## 目的

本文档指导产品研发交付部开展IH Business Process Platform产品开发的开发活动。

## 适用范围

本规范的适用范围包括产品研发交付部的全部工作产品以及相关评审记录。

本规范只在本部门范围内有效，对其他部门没有参考作用。

## 文档读者

本规范的预期读者为智控美信相关领导和参与本部门的所有管理、设计、开发人员，未经智控美信书面允许，请勿扩散。

## 词汇表

iEPP：智控企业结算支付平台（IH Enterprise Payments Platform）

iBPP：业务流程平台（IH Business Process Platform）

## 参考资料

# 环境规范

## 硬件环境

|  |  |
| --- | --- |
| CPU | 2.0+ |
| 内存 | 2G+ |
| 硬盘 |  |
| 网卡 |  |

## 操作系统

|  |  |
| --- | --- |
| 名称 | Windows 系列/Linux |
| 版本 |  |

## 开发工具

|  |  |
| --- | --- |
| IDE | Eclipse Luna |
| 数据库 | Mysql 5.0+/Oracle 10g+ |
| 版本控制 | Subversion 1.6+ |
|  |  |

# 技术规范

## 系统架构

系统采用B/S架构，基于SOA思想，使用Spring Boot等技术，实现业务过程的定制、运行。

## 技术规范

### SOA

SOA(Service-Oriented Architecture)，面向服务架构，它可以根据需求通过网络对松散耦合的粗粒度应用组件进行分布式部署、组合和使用。服务层是SOA的基础，可以直接被应用调用，从而有效控制系统中与软件代理交互的人为依赖性。

SOA是一种粗粒度、松耦合服务架构，服务之间通过简单、精确定义接口进行通讯，不涉及底层编程接口和通讯模型。SOA可以看作是B/S模型、XML/Web Service技术之后的自然延伸。

SOA将能够帮助我们站在一个新的高度理解企业级架构中的各种组件的开发、部署形式，它将帮助企业系统架构者以更迅速、更可靠、更具重用性架构整个业务系统。较之以往，以SOA架构的系统能够更加从容地面对业务的急剧变化。

### Micro Servcie

微服务架构(Micro Service Architecture)是近一段时间在软件体系架构领域里出现的一个新名词。它通过将功能分解到多个独立的服务，以实现对解决方案或者复杂系统的解耦。实际上，微服务本身并没有一个严格的定义。不过从业界的讨论来看，微服务通常有如下几个特征：

* 小，且专注于做一件事情

每个服务都是很小的应用，至于有多小，是一个非常有趣的话题。有人喜欢100行以内，有人赞成1000行以内。数字并不是最重要的。仁者见仁，智者见智，只要团队觉得合适就好。只关注一个业务功能，这一点和我们平常谈论的面向对象原则中的“单一职责原则”类似，每个服务只做一件事情，并且把它做好。

运行在独立的进程中

每个服务都运行在一个独立的操作系统进程中，这意味着不同的服务能被部署到不同的主机上。

* 松耦合

不需要改变依赖，只更改当前服务本身，就可以独立部署。这意味着该服务和其他服务之间在部署和运行上呈现相互独立的状态。分布式部署，有利根据业务需求进行资源分配。

* 轻量级的通信机制

服务和服务之间通过轻量级的机制实现彼此间的通信。所谓轻量级通信机制，通常指基于语言无关、平台无关的这类协议，例如XML、JSON等。

综上所述，微服务架构采用多个服务间互相协作的方式构建传统应用。每个服务独立运行在不同的进程中，服务与服务之间通过轻量的通讯机制交互，并且每个服务可以通过自动化部署方式独立部署。

<http://www.oschina.net/news/70121/microservice>

<http://www.cnblogs.com/loveis715/p/4644266.html>

<http://www.cnblogs.com/lxconan/p/how-to-introduce-micro-service.html>

### Spring

Spring是一个开源框架，它由Rod Johnson创建。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。

　　◆目的：解决企业应用开发的复杂性

　　◆功能：使用基本的JavaBean代替EJB，并提供了更多的企业应用功能

　　◆范围：任何Java应用

　　简单来说，Spring是一个轻量级的控制反转([IoC](http://baike.baidu.com/view/146665.htm" \t "_blank))和面向切面([AOP](http://baike.baidu.com/view/73626.htm))的容器框架。

　　◆轻量——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

　　◆控制反转——Spring通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为IoC与JNDI相反——不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

　　◆面向切面——Spring提供了[面向切面编程](http://baike.baidu.com/view/1865230.htm" \t "_blank)的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

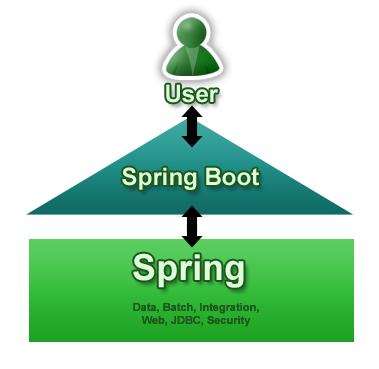
　　◆容器——Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个bean如何被创建——基于一个可配置[原型](http://baike.baidu.com/view/228368.htm" \t "_blank)（prototype），你的bean可以创建一个单独的实例或者每次需要时都生成一个新的实例——以及它们是如何相互关联的。然而，Spring不应该被混同于传统的重量级的EJB容器，它们经常是庞大与笨重的，难以使用。

　　◆框架——Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了你。

所有Spring的这些特征使你能够编写更干净、更可管理、并且更易于测试的代码。它们也为Spring中的各种模块提供了基础支持。

### Spring Boot

Spring Boot 项目旨在简化创建产品级的 Spring 应用和服务。你可通过它来选择不同的 Spring 平台。可创建独立的 Java 应用和 Web 应用，下图显示 Spring Boot 在 Spring 生态中的位置：



该项目主要的目的是：

* 为 Spring 的开发提供了更快更广泛的快速上手
* 使用默认方式实现快速开发
* 提供大多数项目所需的非功能特性，诸如：嵌入式服务器、安全、心跳检查、外部配置等

### ReactJs

### React Native

# 设计规范

## 需求规格说明书规范

## 系统原型设计规范

## 数据库设计规范

## 系统概要设计规范

## 系统详细设计规范

# 编码规范

## 服务端编码规范

### Java编码规范

### Nodejs编码规范

## 前端编码规范

### WEB前端编码规范

### App客户端编码规范

# 开发过程规范

## 开发任务分配、接收

开发任务由产品经理（或模块负责人）通过Redmine进行分配，分配任务时明确定义任务的起止时间、任务明细、责任人等。

开发人员登录Redmine平台，领取任务，并标记为开始执行，执行完成后标记为“已完成”。

注意：

Redmine地址：[http://10.10.69.17/redmine/](http://10.10.69.17/redmine/" \t "_blank)

用户一般有Redmine管理员进行分配，需要登录可自助注册，用户名要求使用姓名全拼，密码不做明确规定。

## 代码版本控制规范

代码通过subversion进行存储，开发人员在本地开发完成后，提交至版本库中，其他开发人员只需进行更新，即可完成协同开发，但在协同开发过程中不可避免会产生代码冲突，因此我们约定svn使用规范如下：

1. 要求严格按照同步->解决冲突->更新-提交顺序完成程序的同步过程
2. 每天早上进行代码更新，第二天上班前完成可用代码的提交
3. 本地私有文件不允许上传，以保证代码库整洁，如Eclipse本地配置文件.classpath,本地编译目录及其文件/WEB-INF/classes

# 测试过程规范

待定