|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **研发生产中心** | **文档编号** | 001[[1]](#footnote-0) | **版本** | A1 | **密级** | 商密A |
| **项目名称** | **智慧健康管理平台** | | | | |
| **项目来源** | 还是威 | | | | |

QR-RD-022(Ver1.2)

**智慧健康管理平台**

**代码结构说明书**

(内部资料 请勿外传)

|  |  |  |  |
| --- | --- | --- | --- |
| **编 写：** | 刘平 | **日 期：** | 2017-09-23 |
| **检 查：** |  | **日 期：** |  |
| **审 核：** |  | **日 期：** |  |
| **批 准：** |  | **日 期：** |  |

**文档变更记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 初版 | 刘平 | 0001 | 2017/9/23 |  |
| 2 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 录

[1 系统架构图 3](#_Toc16050)

[2 技术简介 4](#_Toc20435)

[3 技术配置文件说明 5](#_Toc30533)

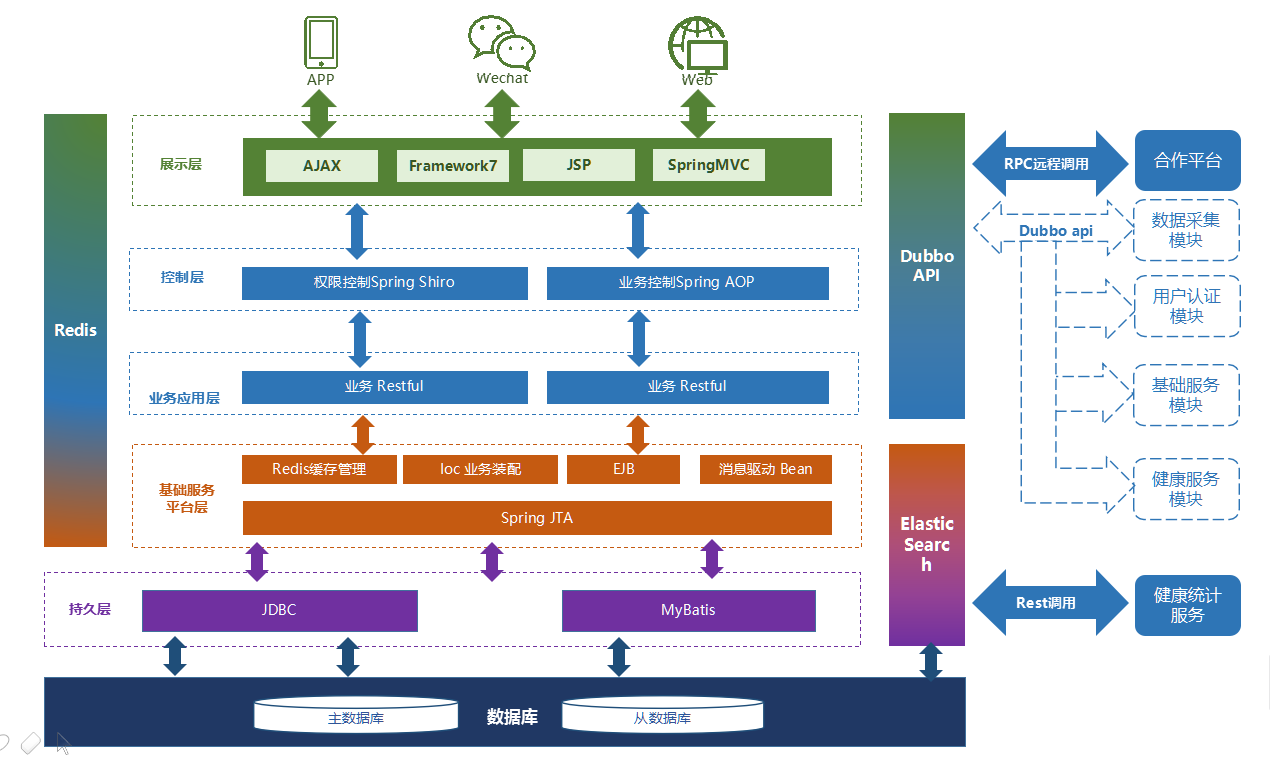
[3.1 平台结构总览 5](#_Toc27872)

[3.2 配置详解 5](#_Toc4839)

[3.2.1 SpringMVC 配置详解 5](#_Toc27710)

[3.2.2 Shiro配置详解 7](#_Toc7721)

# 系统架构图



# 技术简介

使用到的技术包含：SpringMVC、mybatis、Framework7、html、ajax、Redis、ElasticSeach等。

* Spring Web MVC是一种基于Java的实现了Web MVC设计模式的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行职责解耦，基于请求驱动指的就是使用请求-响应模型，框架的目的就是帮助我们简化开发，Spring Web MVC也是要简化我们日常Web开发的。Spring Web MVC也是服务到工作者模式的实现，但进行可优化。前端控制器是DispatcherServlet；应用控制器其实拆为处理器映射器(Handler Mapping)进行处理器管理和视图解析器(View Resolver)进行视图管理；页面控制器/动作/处理器为Controller接口（仅包含ModelAndView handleRequest(request, response) 方法）的实现（也可以是任何的POJO类）；支持本地化（Locale）解析、主题（Theme）解析及文件上传等；提供了非常灵活的数据验证、格式化和数据绑定机制；提供了强大的约定大于配置（惯例优先原则）的契约式编程支持。
* Mybatis是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。
* Framework7 是一款免费开源的移动web开发框架(类似的知名框架有Jquery Mobile,sencha),用以开发IOS7风格、体验的混合移动应用和web应用。如果你需要，它还是一款绝好的移动应用原型框架，用以展示应用的功能细节。Framework7框架的主要目的是使得开发者能够简单、快速的使用HTML,Javascript,CSS开发IOS风格的webapp和混合本地应用。
* Ajax即“Asynchronous Javascript And XML”（异步JavaScript和XML），是指一种创建交互式[网页](https://baike.baidu.com/item/%E7%BD%91%E9%A1%B5" \t "https://baike.baidu.com/item/ajax/_blank)应用的网页开发技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。
* Redis是一个开源的使用ANSI [C语言](https://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/_blank)编写、支持网络、可基于内存亦可持久化的日志型、Key-Value[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93" \t "https://baike.baidu.com/item/_blank)，并提供多种语言的API。在此系统中只用于缓存数据库。
* ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于[云计算](https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97" \t "https://baike.baidu.com/item/_blank)中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

# 技术配置文件说明

## 平台结构总览



部署时，api与app都会编译成jar包，放在admin.war下。

## 配置详解



其详细配置，如下只列出两个，其它的可在代码里查看。

### SpringMVC 配置详解

xml文件里每个节点都有详细的注释，直接查看源码，例如base-servlet.xml文件：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:task=*"http://www.springframework.org/schema/task"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-4.3.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-4.3.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.3.xsd*  *http://www.springframework.org/schema/task*  *http://www.springframework.org/schema/task/spring-task-4.3.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd"*>  <!-- 自动扫描且只扫描@Controller -->  <context:component-scan base-package=*"com.hesvit"* use-default-filters=*"false"*>  <context:include-filter type=*"annotation"* expression=*"org.springframework.stereotype.Controller"*/>  <context:include-filter type=*"annotation"* expression=*"org.springframework.web.bind.annotation.ControllerAdvice"*/>  </context:component-scan>    <!-- 参数编码转换 -->  <mvc:annotation-driven>  <mvc:message-converters register-defaults=*"true"*>  <bean class=*"org.springframework.http.converter.StringHttpMessageConverter"*>  <constructor-arg value=*"UTF-8"* />  </bean>  <bean class=*"org.springframework.http.converter.json.MappingJackson2HttpMessageConverter"*>  <property name=*"prettyPrint"* value=*"true"*/>  </bean>  </mvc:message-converters>  </mvc:annotation-driven>    <!-- 请求根时，自动转发到index的action -->  <mvc:view-controller path=*"/"* view-name=*"forward:/index"*/>    <!-- REST中根据URL后缀自动判定Content-Type及相应的View -->  <bean id=*"contentNegotiationManager"* class=*"org.springframework.web.accept.ContentNegotiationManagerFactoryBean"*>  <property name=*"ignoreAcceptHeader"* value=*"true"* />  <property name=*"defaultContentType"* value=*"application/json"* />  <property name=*"mediaTypes"* >  <value>  json=application/json  xml=application/xml  </value>  </property>  </bean>    <!-- 定义JSP文件的位置 -->  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/"*/>  <property name=*"suffix"* value=*".html"*/>  </bean>    <!-- 容器默认的DefaultServletHandler处理 所有静态内容与无RequestMapping处理的URL-->  <mvc:default-servlet-handler/>    <!-- Shiro注解支持 -->  <aop:aspectj-autoproxy proxy-target-class=*"true"* />  <bean class=*"org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor"*>  <property name=*"securityManager"* ref=*"securityManager"*/>  </bean>  </beans> |

### Shiro配置详解

base-shiro.xml里的每个节点也有详细的解释，例如：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:util=*"http://www.springframework.org/schema/util"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/util*  *http://www.springframework.org/schema/util/spring-util.xsd"*>  <!-- 启用shrio授权注解拦截方式 -->  <bean id=*"shiroFilter"* class=*"org.apache.shiro.spring.web.ShiroFilterFactoryBean"*>  <property name=*"securityManager"* ref=*"securityManager"* /><!-- 装配 securityManager -->  <property name=*"loginUrl"* value=*"/welcome"* /><!-- 配置登陆页面 -->  <property name=*"successUrl"* value=*"/${shiro.login.success.url}"*/><!-- 登陆成功后的一面 -->  <property name=*"unauthorizedUrl"* value=*"/welcome"* /><!-- 未授权跳转页面 -->  <!-- 具体配置需要拦截哪些 URL, 以及访问对应的 URL 时使用 Shiro 的什么 Filter 进行拦截 -->  <property name=*"filters"*>  <util:map>  <entry key=*"hesvitAuthc"* value-ref=*"hesvitAuthcFilter"* />  <entry key=*"hesivtUser"* value-ref=*"hesvitUserFilter"* />  <entry key=*"commonAuthc"* value-ref=*"commonAuthcFilter"* />  <entry key=*"commonUser"* value-ref=*"commonUserFilter"* />  <entry key=*"logout"* value-ref=*"logoutFilter"* />  </util:map>  </property>  <!--anon匿名 authc登录认证 user用户已登录 logout退出filter -->  <property name=*"filterChainDefinitions"*>  <value>  /public/\*\* = anon  /demo/\*\* = anon  /app/\*\* = anon  /authimage = anon  /welcome = anon  /${shiro.hesvit.login}/login = hesvitAuthc  /${shiro.hesvit.login}/login/\*\* = anon  /${shiro.common.login}/login = commonAuthc  /${shiro.common.login}/login/\*\* = anon  /logout = logout  /\*\* = hesivtUser  </value>  </property>  </bean>  <!-- Shiro Filter -->  <!-- 平台用户授权拦截 -->  <bean id=*"hesvitAuthcFilter"* class=*"com.hesvit.common.security.IntelAuthenticationFilter"* >  <property name=*"loginUrl"* value=*"/${shiro.hesvit.login}${shiro.login.url}"* />  <property name=*"successUrl"* value=*"/${shiro.login.success.url}"* />  </bean>  <!-- 平台用户拦截 -->  <bean id=*"hesvitUserFilter"* class=*"com.hesvit.common.security.IntelUserFilter"* >  <property name=*"loginUrl"* value=*"/welcome"* />  </bean>  <!-- 机构用户授权拦截 -->  <bean id=*"commonAuthcFilter"* class=*"com.hesvit.common.security.IntelAuthenticationFilter"* >  <property name=*"loginUrl"* value=*"/${shiro.common.login}${shiro.login.url}"* />  <property name=*"successUrl"* value=*"/${shiro.login.success.url}"* />  </bean>  <!-- 机构用户拦截 -->  <bean id=*"commonUserFilter"* class=*"com.hesvit.common.security.IntelUserFilter"* >  <property name=*"loginUrl"* value=*"/${shiro.common.login}${shiro.login.url}"* />  </bean>  <!-- 角色拦截 -->  <bean id=*"roles"* class=*"com.hesvit.common.security.IntelRolesAuthorizationFilter"* />  <!-- 权限拦截 -->  <bean id=*"perms"* class=*"com.hesvit.common.security.IntelPermissionsAuthorizationFilter"* />  <!-- 退出登录过滤器 -->  <bean id=*"logoutFilter"* class=*"com.hesvit.common.security.IntelLogoutFilter"*>  <property name=*"redirectUrl"* value=*"/welcome"* />  </bean>  <!-- 配置 Shiro 的 SecurityManager Bean -->  <bean id=*"securityManager"* class=*"org.apache.shiro.web.mgt.DefaultWebSecurityManager"*>  <property name=*"realm"* ref=*"userRealm"* />  <property name=*"cacheManager"* ref=*"shiroCacheManager"* />  <property name=*"sessionManager"* ref=*"sessionManager"* />  <property name=*"rememberMeManager"* ref=*"rememberMeManager"* />  </bean>  <!-- 用户权限验证 -->  <bean id=*"userRealm"* class=*"com.hesvit.common.security.IntelUserRealm"*> </bean>    <!-- 会话管理器 -->  <bean id=*"sessionManager"* class=*"org.apache.shiro.web.session.mgt.DefaultWebSessionManager"*>  <property name=*"globalSessionTimeout"* value=*"${shiro.session.globalSessionTimeout}"* />  <property name=*"sessionFactory"* ref=*"sessionFactory1"* />  <property name=*"sessionDAO"* ref=*"sessionDAO"* />  <property name=*"deleteInvalidSessions"* value=*"true"* />  <property name=*"sessionValidationInterval"* value=*"${shiro.session.validation.interval}"* />  <property name=*"sessionValidationSchedulerEnabled"* value=*"false"* />  <!-- <property name="sessionValidationScheduler" ref="sessionValidationScheduler" /> -->  <property name=*"sessionIdCookieEnabled"* value=*"true"* />  <property name=*"sessionIdCookie"* ref=*"sessionIdCookie"* />  </bean>  <!-- session cookie信息 -->  <bean id=*"sessionIdCookie"* class=*"org.apache.shiro.web.servlet.SimpleCookie"*>  <constructor-arg value=*"${shiro.intel.cookie.name}"* />  <property name=*"domain"* value=*"${shiro.intel.cookie.domain}"* />  <property name=*"path"* value=*"${shiro.intel.cookie.path}"* />  <property name=*"httpOnly"* value=*"${shiro.intel.cookie.httpOnly}"* />  <property name=*"maxAge"* value=*"${shiro.intel.cookie.maxAge}"* />  </bean>  <!-- Remembered vs Authenticated http://www.ituring.com.cn/article/287 -->  <bean id=*"rememberMeManager"* class=*"org.apache.shiro.web.mgt.CookieRememberMeManager"*>  <!-- rememberme cookie加密的密钥 建议每个项目都不一样 默认AES算法 密钥长度（128 256 512 位） -->  <property name=*"cipherKey"* value=*"#{T(org.apache.shiro.codec.Base64).decode('${shiro.intel.rememeberMe.cookie.base64.cipherKey}')}"* />  <property name=*"cookie"* ref=*"rememberMeCookie"* />  </bean>  <!---cookie -->  <bean id=*"rememberMeCookie"* class=*"org.apache.shiro.web.servlet.SimpleCookie"*>  <constructor-arg value=*"${shiro.intel.rememeberMe.cookie.name}"* />  <property name=*"domain"* value=*"${shiro.intel.cookie.domain}"* />  <property name=*"path"* value=*"${shiro.intel.cookie.path}"* />  <property name=*"httpOnly"* value=*"${shiro.intel.cookie.httpOnly}"* />  <property name=*"maxAge"* value=*"${shiro.intel.rememeberMe.cookie.maxAge}"* />  </bean>  <!-- 会话验证调度 org.apache.shiro.session.mgt.quartz.Quartz2SessionValidationScheduler -->  <!-- com.hesvit.common.task.SpringSessionValidationScheduler -->  <!-- <bean id="sessionValidationScheduler" class="com.hesvit.common.task.Quartz2SessionValidationScheduler"> -->  <!-- <property name="sessionValidationInterval" value="${shiro.session.validation.interval}" /> -->  <!-- <property name="sessionManager" ref="sessionManager" /> -->  <!-- <property name="scheduler" ref="scheduler" /> -->  <!-- </bean> -->    <!-- Shiro Session Factory -->  <bean id=*"sessionFactory1"* class=*"org.apache.shiro.session.mgt.SimpleSessionFactory"* />  <!-- uid(session id) 生成策略 -->  <bean id=*"sessionIdGenerator"* class=*"org.apache.shiro.session.mgt.eis.JavaUuidSessionIdGenerator"* />  <!-- 会话保持的DAO -->  <bean id=*"sessionDAO"* class=*"org.apache.shiro.session.mgt.eis.EnterpriseCacheSessionDAO"*>  <property name=*"sessionIdGenerator"* ref=*"sessionIdGenerator"* />  <property name=*"activeSessionsCacheName"* value=*"${shiro.active.session.cacheName}"* />  </bean>  <!-- shiro缓存管理 -->  <bean id=*"shiroCacheManager"* class=*"org.apache.shiro.cache.MemoryConstrainedCacheManager"*></bean>  <!-- Enable Shiro Annotations for Spring-configured beans. Only run after the lifecycleBeanProcessor has run: -->  <!-- 保证实现了Shiro内部lifecycle函数的bean执行 -->  <bean id=*"lifecycleBeanPostProcessor"* class=*"org.apache.shiro.spring.LifecycleBeanPostProcessor"* />  <!-- Spring远程安全确保每个远程方法调用都与一个负责安全验证的Subject绑定 -->  <bean id=*"secureRemoteInvocationExecutor"* class=*"org.apache.shiro.spring.remoting.SecureRemoteInvocationExecutor"*>  <property name=*"securityManager"* ref=*"securityManager"* />  </bean>    </beans> |

1. [↑](#footnote-ref-0)