

**NHIS\_SpringBoot版开发说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件状态：  【√】草稿  【 】正式发布  【 】正在修改 | 文件编号 |  | 机密等级 |  |
| 负责人 |  | 文档版本 |  |
| 审核 |  | 生效日期 |  |

**江苏振邦智慧城市信息系统有限公司**

电话：0519-89606206 传真：0519-89606203

地址：江苏省常州市武进区牛塘工业园区创新路88号

邮编：213168

网址：www.zebone. cn

目录

[1. 概述 3](#_Toc79673477)

[1.1. 文档目的 3](#_Toc79673478)

[1.2. 阅读说明 3](#_Toc79673479)

[1.3. 环境要求 3](#_Toc79673480)

[1.4. 环境配置 3](#_Toc79673481)

[1.4.1. Java环境变量配置 3](#_Toc79673482)

[1.4.2. Maven环境配置说明 4](#_Toc79673483)

[1.5. 研发环境启动 7](#_Toc79673484)

~~[1.5.1.](#_Toc79673485)~~~~[安装jar包](#_Toc79673485)~~ [7](#_Toc79673485)

[1.5.2. 导入工程 7](#_Toc79673486)

[1.6. 常见问题 10](#_Toc79673487)

[1.6.1. 依赖包找不到 10](#_Toc79673488)

[1.6.2. 编译异常 10](#_Toc79673489)

[1.6.3. 日志文件找不到 10](#_Toc79673490)

[1.6.4. 启动后404 11](#_Toc79673491)

[1.7. 打包部署 11](#_Toc79673492)

[1.7.1. 环境准备 11](#_Toc79673493)

[1.7.2. 打包 11](#_Toc79673494)

[1.7.3. 启动 13](#_Toc79673495)

[2. 结构说明 13](#_Toc79673496)

[2.1. 整体结构说明 13](#_Toc79673497)

[2.2. 细节说明： 14](#_Toc79673498)

[2.2.1. 消息配置 14](#_Toc79673499)

[2.2.2. Webservice配置 14](#_Toc79673500)

[2.2.3. 其他 15](#_Toc79673501)

[3. 升级内容 15](#_Toc79673502)

[3.1.1. Spring系列升级 15](#_Toc79673503)

[3.1.2. 其他核心组件升级 18](#_Toc79673504)

[4. 其他说明 19](#_Toc79673505)

[4.1. 升级后优点 19](#_Toc79673506)

[4.2. 存在的风险 20](#_Toc79673507)

[4.3. 升级改造过程 20](#_Toc79673508)

# 概述

## 文档目的

指导开发人员搭建springboot环境，熟悉关键变更内容。

## 阅读说明

针对日常开发，开发人员只需要关注前4章节内容。开发过程中，以业务实现为核心，在满足此目标的前提下，尽可能的去使用新框架带来的新特性去快速、稳定、优雅的编码，减少代码的臃肿性。

## 环境要求

1. JDK版本，Redis版本和之前一样即可
2. Tomcat 版本>=9.0.40 可以到官网直接下载9.0.50即可
3. Maven 版本>=Maven 3.6.0
4. *可要能连接互联网，已经搭建中央仓库，需要vpn，也可以从公网下载依赖包*

## 环境配置

### Java环境变量配置

同以前开发配置。

### Maven环境配置说明

#### 环境变量配置

1. 下载好之后放到某个目录，比如这里放到了D:\apache-maven-3.6.0，然后配置环境变量MAVEN\_HOME:D:\apache-maven-3.6.0；
2. path中追加%MAVEN\_HOME%\bin\
3. Cmd控制台执行mvn -version正确输入内容就可以

#### 其他配置

1. 配置私有maven仓库服务节maven**(仅使用团队自己搭建的maven仓库时配置)**

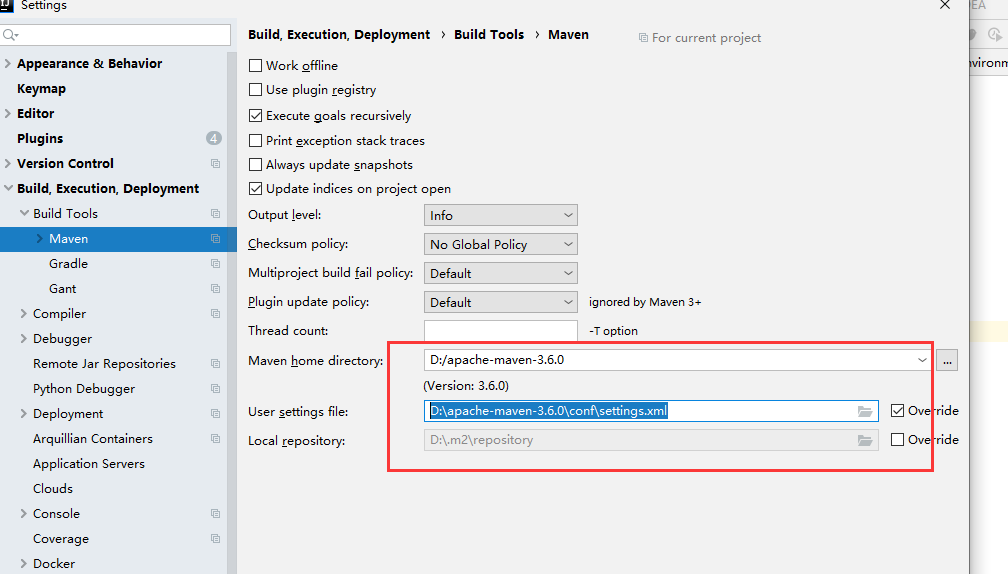
|  |  |
| --- | --- |
| <server>  <id>maven-nexus</id>  <username>zbRep</username>  <password>123456</password>  </server> | maven中的目录conf\settings.xml servers节点中加入即可。  其他想要配置全局私服地址的，和下面的配置阿里地址一样，**id**不同即可。 |

1. 配置maven本地仓库地址和镜像地址：  
    找到maven中的目录conf\settings.xml加入仓库位置：（也可以不改，默认放在用户目录.m2文件夹中）

|  |
| --- |
| 本地仓库地址默认是用户目录.m2文件夹，可以修改为其他目录： |
| **mirrors中注释掉其他的，加入阿里镜像地址：**  <mirror>  <id>maven-nexus</id>  <mirrorOf>repositoryId</mirrorOf>  <name>our mirror.</name>  <url>http://192.168.100.224:8081/repository/zb-repositories/</url>  </mirror>  <mirror>  <id>alimaven</id>  <mirrorOf>central</mirrorOf>  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>  </mirror> |
| **加入profile，方便本地编译**   <profile>        <id>jdk-1.8</id>        <activation>        <activeByDefault>true</activeByDefault>        <jdk>1.8</jdk>        </activation>        <properties>        <maven.compiler.source>1.8</maven.compiler.source>        <maven.compiler.target>1.8</maven.compiler.target>        <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>        </properties>        <repositories>        <repository>        <id>maven-nexus</id>        <url>http://192.168.100.224:8081/repository/zb-repositories/</url>        <snapshotPolicy>always</snapshotPolicy>        </repository>        </repositories>        </profile> |

#### Idea环境配置

指定自己的maven配置目录即可



## 研发环境启动

### ~~安装jar包~~

(因部分包在阿里仓库中无法找到，在使用阿里在线maven仓库时配置，使用私有仓库无需配置)

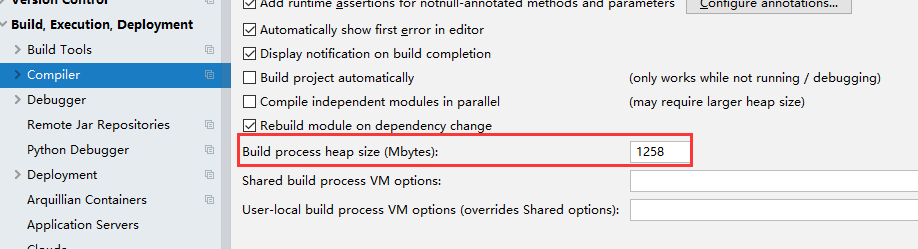
|  |
| --- |
| 进入nhis-doc/extLib目录执行mvnInstallExtjar.bat 如果一切正常，所有的窗口都应该展示成功：    \*\*注意，使用cmd,别用powershell、Terminal这些，避免不必要的异常 |
| 如果是使用Mac系统，只需要将bat命令里面的mvn打头的拿出来自己到上面的目录去执行即可 |
| 后面如果我们自建了maven仓库，将不用这么麻烦，都从中央仓库下载即可。 |

### 导入工程

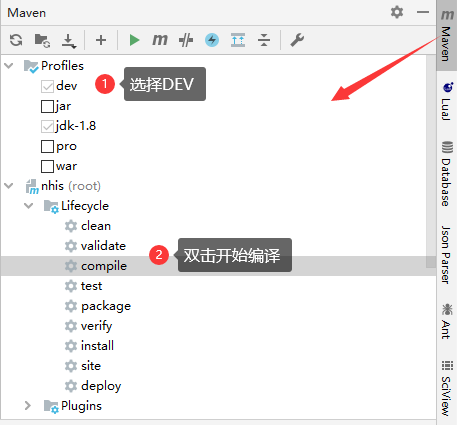
#### Idea导入

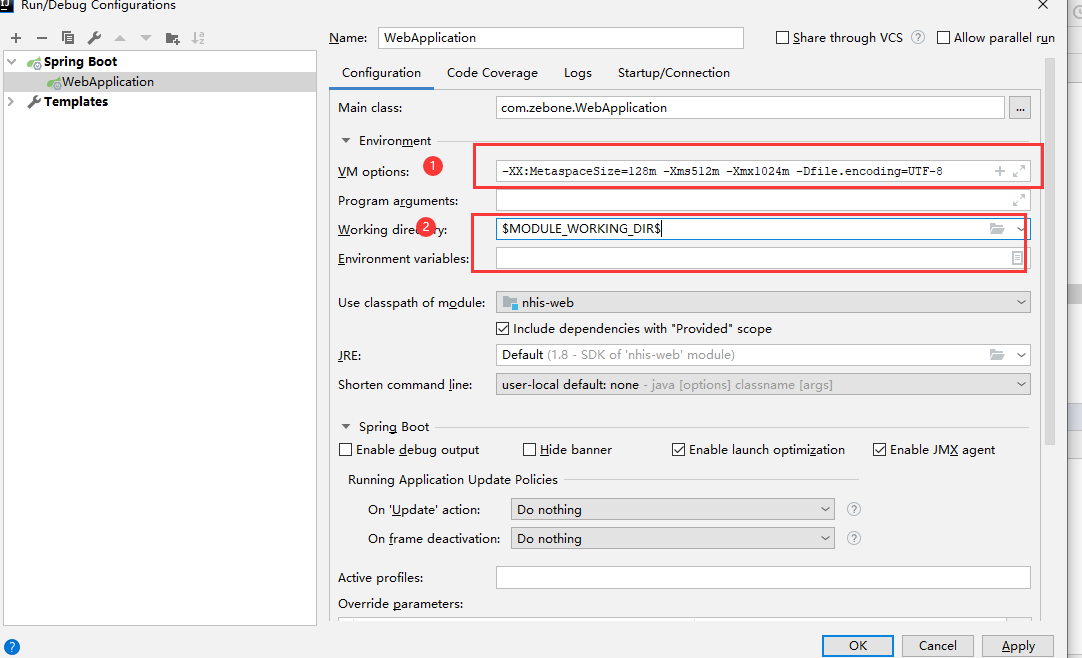
1. 导入和基本配置
2. 和以前一样File>open...选择项目所在目录即可，注意选择到外层目录即可。
3. 导入成功后，再次检查java版本设置是否为1.8
4. Maven会自动下载依赖包，如果没有下载，刷新右侧的maven图标

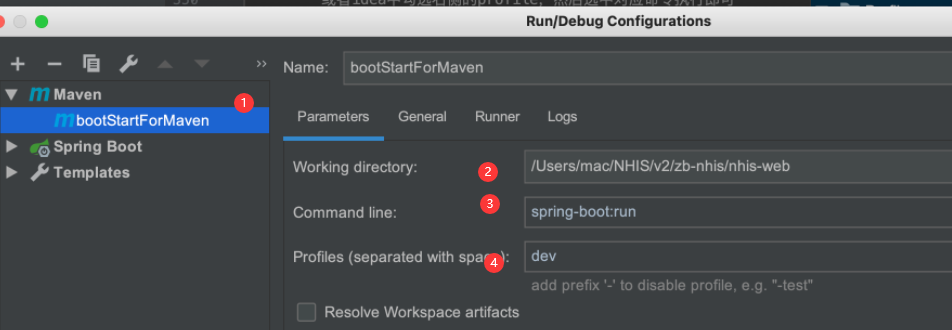
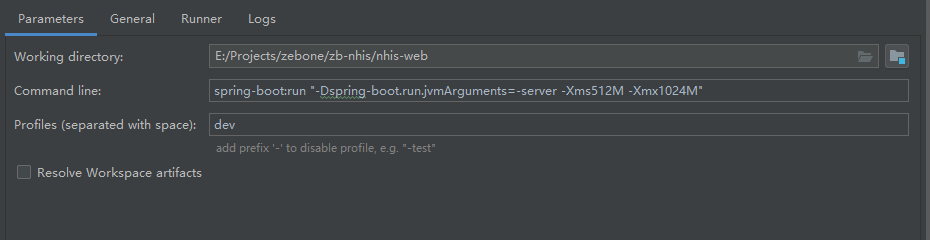
2. 编译和启动

A)首次编译很耗时，也很耗费内存，建议调大内存：

1. 编译：



C)启动：到类WebApplication直接运行，或者debug运行  


D)其他启动方式：idea中使用maven启动：注意Runner中设置好jdk  
  


(1)使用maven命令启动  
mvn spring-boot:run -Pdev -Dspring-boot.run.jvmArguments="-Xms512m -Xmx512m"

(2)打包后丢到外部Tomcat启动，参见[打包](#_打包)启动

#### 其他IDE

比如Eclipse等，需要自行安装插件即可，其他配置与IDEA类似。

#### 引入第三方包

**暂不允许开发人员私自引入第三包，如需引入请联系产品开发相关负责人。**

## 常见问题

### 依赖包找不到

查看本地maven仓库是否已经下载或者安装对应的jar包；执行maven clean命令，刷新maven组件；有时候包在，只是有红色提示，关闭打开IDE即可，也可以删掉仓库里对应的包，重新compile会在下载。

### 编译异常

如果是jdk的内置包，很有可能你指定的jdk版本有问题。如果是其他三方包，看看maven是否引入成功。

### 日志文件找不到

日志文件的位置，是在logback-dev/pro中指定的，默认开发环境指定了D:\logs 那么所有的日志在这个目录下。

### 启动后404

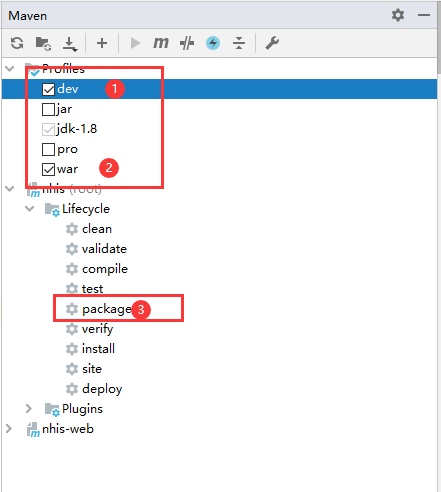
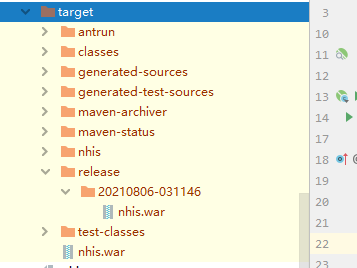
看上面，是不是设置好了idea的working direcotry。看看内置\*jasper包是否存在。

## 打包部署

### 环境准备

不需要maven。Jdk和Tomcat版本说明同上即可。

### 打包

1. 开发环境打war包，比如要使用的是dev，勾选dev、war，执行package；一切正常后会在target目录生成war包。同时target/release中会有按照时间戳生成的nhis.war，两个一样的。  
     
   生成后类似这样：  
   
2. 打jar包的姿势相似，勾选dev和jar即可，运行使用jar命令即可

### 启动

War启动和以前的方式一模一样，不再赘述。

# 结构说明

## 整体结构说明

|  |  |
| --- | --- |
|  | 和以前Springmvc工程一样的目录： |
| etc |
| java |
| webapp |
|  |
| 日志配置：  etc/logback-dev.xml配置开发环境的日志输出  etc/logback-pro.xml：对应生产配置 |

升级之后特有目录：

resources

/static:放的静态内容，我们没有前后分离之前都放在这里。

/templates:放的模板视图内容，比如现在放置的是freemarker的模板内容。

application.yml:主配置文件，不区分是开发环境还是生产环境，都得有。

application-dev.yml:开发环境的配置

application-pro.yml:生产环境的配置

application-ws.yml:等于以前的applicationContext-cxf的配置。各项目按照自己需要配置要发布的webservice

## 细节说明：

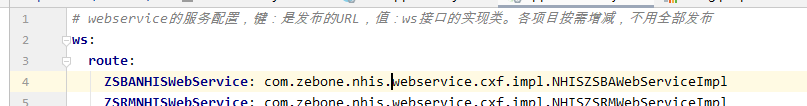
### 消息配置

以前的Mina接受消息配置是在applicationContext-mina.xml。现在config.properties中加入两行配置：  


### Webservice配置

以前WS配置实在applicationContext-cxf.xml。现在开发好接口后（和以前一模一样），在application-ws.yml中配置，加入需要发布的URL名称和对应的实现类：

比如第一个发布后就是http://localhost:8888/static/webservice/ZSBANHISWebservice?wsdl



### 其他

没有特别说明的，开发方式和以前一模一样，只是在使用一些高级功能，和Springboot交互是会有区别，具体问题具体实现即可。

# 升级内容

### Spring系列升级

#### 主版本说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Old Version** | **New Version** | **Detail** | **Note** |
| Spring | 3.2.10 | 5.3.8 | Spring-\* 依赖所有内容同时升级 | 主版本由Springboot控制，Spring版本由boot决定。 |
| Springboot |  | 2.5.2 | Spring-boot\* 依赖的所有内容都是相同的版本  包括内置的\*starter都会是相同版本  三方扩展的按照版本依赖选择匹配 |

#### 细节说明

|  |  |
| --- | --- |
| **Name** | **Note** |
| Spring-jdbc | 1. 移除了queryForInt,queryForLong等的查询。全部使用queryForObject+泛型来查询 |
| Spring-MVC | 1. 加入@RestController注解(@Controller和@ResponseBody注解的合体) (>=4.0) 2. 新增AsyncRestTemplate类，开发REST客户端时允许非阻塞异步支持(>=4.0) 3. [Jackson的@JsonView直接支持@ResponseBody和ResponseEntity控制器方法(4.1)https://www.jianshu.com/p/dab9dae5bf20](mailto:Jackson的@JsonView直接支持@ResponseBody和ResponseEntity控制器方法(4.1)https:/www.jianshu.com/p/dab9dae5bf20) 4. 内置对CORS的支持(4.2) 5. 新的@GetMapping，@PostMapping，@PutMapping，@DeleteMapping和@PatchMapping @RequestMapping的组合注释(4.3) 6. 新的@SessionAttribute @RequestAttribute注释用于访问session、request属性(4.3) 7. 缓存的改进等：支持多个cacheManager,并且可以使用注解灵活配置，诸如实现jvm层面的缓存、同时基于Redis、EhCache等多个缓存选项 8. SpEL（Spring Expression Language）表达式的使用等<https://docs.spring.io/spring-integration/docs/5.3.8.RELEASE/reference/html/spel.html> 9. 使用 Kotlin 进行函数式编程。Spring Framework 5.0 引入了对 JetBrains Kotlin 语言的支持 >5 10. 响应式编程模型 >5 11. 额外库支持:核心的比如OkHttp 3.x+ JDBC 4.0+ EhCache 2.10+ / 3.0 GA  * [What's New in Spring Framework 4.x](https://github.com/spring-projects/spring-framework/wiki/What's-New-in-the-Spring-Framework" \l "whats-new-in-spring-framework-4x) * [What's New in Spring Framework 5.x](https://github.com/spring-projects/spring-framework/wiki/What's-New-in-the-Spring-Framework" \l "whats-new-in-spring-framework-5x) |
| Springboot | 1.自动配置：与SpringMVC工程相比，依据约定大于配置的理念，使用spring-boot-autoconfiguration系列来实现自动配置。 2.简化maven配置，parent这种方式，一站式引入需要的各种依赖 3.默认的配置，基于JavaConfig的自定义配置，各种@EnableAutoConfiguration、@Conditional等  4.内置依赖：默认提供基础的依赖包，以及最佳适配包的版本定义，避免了复杂的依赖，不清楚应该引入哪个版本的包等状况。  5.内置监控：actuator，actuator主要是对springboot应用程序进行监控和管理 |

### 其他核心组件升级

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Old Version** | **New Version** | **Note** |
| mybatis | 3.1.1 | 3.5.5 | 版本升级后的常用的新特性：   1. 支持从多个结果集填充结果： 2. 支持Java 8中添加的新日期和时间API (JSR-310)类。 3. @Select 可以返回对象数组 4. 允许用户在占位符中设置默认值${属性:默认值}。需要开启 5. 支持返回枚举 6. 在包含的SQL片段的属性值中替换变量   支持java.util.Optional作为映射器方法的返回类型。  @Select("select \* from users where id = #{id}")  Optional<User> findOne(Integer id);   1. 支持多行插入 2. 支持包含<sql />的CDATA变量替换 3. 在包含的SQL片段的属性值中替换变量  例如：不用再同样的语句复制多条，可以通过include标签配合变量替换完美解决。这样修改只需要改一个地方 |
| 其他 |  |  | cxf,shiro，Quartz，apache-common\*,jackson,等都跟着升级了，兼容以前的内容。 |

# 其他说明

## 升级后优点

1. 首先，仅进行同系列的技术体系升级，业务内容，外部接口等无感知。其次，目前从事Java开发的程序员大多使用Springboot框架，持续使用过时技术不利于人才引进。最后，广泛使用的技术，出现问题，开发人员通过百度等方式，网上能找到解决方案，如果太久甚至已经使用停止维护的，可能就找不到解决方法。
2. 提升程序可扩展性，便于后期更快、更稳定的持续集成
3. 升级后一些特性，会在某些场景发挥很大作用，简化开发，提高编码效率，编码质量

## 存在的风险

1. 一些太老久的包找不到，无法升级，只能沿用，在集成过程中只能做到不报错，可能运行时，才会发现问题
2. 为了适应以前的开发习惯，我们已经尽可能的做到业务开发平常和之前一模一样，也就是说以前的姿势仍然适用。包括一些过期类等，但这个也许未来某个版本会被删除。
3. 包冲突问题：如果发生大概率开发、测试阶段就会发现。也有一定几率可能没有测到，某个功能点运行时触发异常

## 升级改造过程

1. 先搭建出一个纯净的maven+Springboot项目，然后整合SSM框架，陆续整合Shiro、Mina等其他大的组件进去（没有放入任何业务代码）。放入的姿势保证和原来的几乎一样，避免配置不同引发的问题（主要依赖于各种applicationContext-xxx.xml配置文件）。
2. 第一步逐步整合完成后，尝试启动项目，进行验证集成效果
3. 放入部分业务代码，主要是底层的platform，zb-framework包的内容，依据错误内容，引入相关jar包。注意版本匹配问题
4. 将我们的etc/config放入，整合并解决旧的使用姿势问题
5. 放入全部业务代码，解决编译异常，如上解决jar包问题。（有可能升级后有些类没了，或者方法变了，需要手动改造）
6. 解决包冲突问题，使用maven命令查看依赖树，修正依赖问题，重新启动测试
7. 逐步解决编译异常，运行时异常，直到能完整启动。
8. 接下来自行测试我们的各种组件：mvc、Mybatis、事务管理、Shiro安全管理、Mina消息、Webservice接口等等