SpringMVC+Spring+Mybatis环境搭建

|  |  |
| --- | --- |
|  | 🢂 内容概览 |
|  | Why：此文档用来做什么？它存在的意义是什么？为解决什么问题？   |  | | --- | |  |   What：当前包含了那些内容？   |  | | --- | |  |   How：此文档应如何参考？   |  | | --- | |  |   Who：此文档适用于那些人员阅读参考？   |  | | --- | |  | |

目录

[1 Maven配置 3](#_Toc437799411)

[1.1 基本属性 3](#_Toc437799412)

[1.2 Resource 3](#_Toc437799413)

[1.3 Plugin 3](#_Toc437799414)

[1.3.1 Maven-compiler 3](#_Toc437799415)

[1.3.2 Maven-surefire 4](#_Toc437799416)

[1.4 Dependency 4](#_Toc437799417)

[1.4.1 Junit4 4](#_Toc437799418)

[1.4.2 Java web 4](#_Toc437799419)

[1.4.3 Logback 5](#_Toc437799420)

[1.4.4 Spring 5](#_Toc437799421)

[1.4.5 Mybatis 6](#_Toc437799422)

[1.4.6 Mysql连接器 7](#_Toc437799423)

[1.4.7 Druid数据源 7](#_Toc437799424)

[1.4.8 Redis 7](#_Toc437799425)

[1.4.9 Validator 7](#_Toc437799426)

[1.4.10 Google-collections 7](#_Toc437799427)

[2 Web.xml配置 7](#_Toc437799428)

[2.1 Spring配置 8](#_Toc437799429)

[2.2 字符编码过滤器 8](#_Toc437799430)

[2.3 Spring mvc核心控制器 8](#_Toc437799431)

[2.4 Druid监控 9](#_Toc437799432)

# Maven配置

## 基本属性

|  |
| --- |
| <!-- pom文档模型版本 --> <modelVersion>4.0.0</modelVersion>  <!-- 项目坐标 --> <!-- 当前项目groupId --> <groupId>com.fsindustry.cime</groupId> <!-- 当前项目artifactId --> <artifactId>cimebm</artifactId> <!-- 当前项目版本号 --> <version>0.0.1-SNAPSHOT</version> <!-- 项目打包方式 --> <packaging>war</packaging> <!-- 项目名称 --> <name>cimebm</name> <!-- 项目主页 --> <url>http://www.fsindustry.cn</url> |

## Resource

|  |
| --- |
| <!--配置Maven 对resource文件 过滤 --> <resources>  <resource>  <directory>src/main/resources</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>true</filtering>  </resource>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>true</filtering>  </resource> </resources> |

## Plugin

### Maven-compiler

Maven编译插件

|  |
| --- |
| <!-- Maven插件 配置 --> <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>${plugin.maven-compiler}</version>  <configuration>  <source>${project.build.jdk}</source>  <target>${project.build.jdk}</target>  <encoding>${project.build.sourceEncoding}</encoding>  </configuration> </plugin> |

### Maven-surefire

Maven测试插件，其实大部分时间内，只要你的测试 类遵循通用的命令约定（以Test结尾、以TestCase结尾、或者以Test开头），就几乎不用知晓该插件的存在。然而在当你想要跳过测试、排除某些 测试类、或者使用一些TestNG特性的时候，了解maven-surefire-plugin的一些配置选项就很有用了。例如 **mvn test -Dtest=FooTest** 这样一条命令的效果是仅运行FooTest测试类，这是通过控制maven-surefire-plugin的test参数实现的。

|  |
| --- |
| <!-- maven测试插件 --> <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <version>${plugin.maven-surefire}</version>  <configuration>  <skipTests>${plugin.maven-surefire.skipTests}</skipTests>  </configuration> </plugin> |

## Dependency

### Junit4

|  |
| --- |
| <!-- junit4 --> <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>${junit.version}</version> </dependency> |

### Java web

|  |
| --- |
| <!-- servlet api --> <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>${javax.servlet.api}</version>  <scope>provided</scope> </dependency>  <!-- jstl --> <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>${jstl.version}</version> </dependency> |

### Logback

|  |
| --- |
| <!--logback--> <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>${slf4j.api.version}</version> </dependency> <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-core</artifactId>  <version>${logback.core.version}</version> </dependency> <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-classic</artifactId>  <version>${logback.classic.version}</version> </dependency> |

### Spring

Spring依赖包括spring和aspectj两部分

|  |
| --- |
| <!-- spring framework --> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-expression</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-oxm</artifactId>  <version>4.0.2.RELEASE</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>${spring.version}</version> </dependency> <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version> </dependency>  <!-- aspectj --> <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjrt</artifactId>  <version>${aspectj}</version> </dependency> <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjweaver</artifactId>  <version>${aspectj}</version> </dependency> |

### Mybatis

|  |
| --- |
| <!-- mybatis --> <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>${mybatis.version}</version> </dependency> <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>${mybatis.spring.version}</version> </dependency> <!-- mybatis分页插件 --> <dependency>  <groupId>com.github.pagehelper</groupId>  <artifactId>pagehelper</artifactId>  <version>${pagehelper.version}</version> </dependency> <dependency>  <groupId>com.github.jsqlparser</groupId>  <artifactId>jsqlparser</artifactId>  <version>${jsqlparser.version}</version> </dependency> |

### Mysql连接器

|  |
| --- |
| <!-- mysql --> <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>${mysql.connector.version}</version> </dependency> |

### Druid数据源

|  |
| --- |
| <!-- druid数据源 --> <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>${druid.version}</version> </dependency> |

### Redis

|  |
| --- |
| <!-- redis --> <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>${jedis.version}</version> </dependency> |

### Validator

|  |
| --- |
| <!-- hibernate-validator --> <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-validator</artifactId>  <version>${hibernate.validator.version}</version> </dependency> |

### Google-collections

|  |
| --- |
| <!-- google collections --> <dependency>  <groupId>com.google.collections</groupId>  <artifactId>google-collections</artifactId>  <version>${google.collections.version}</version> </dependency> |

# Web.xml配置

## Spring配置

|  |
| --- |
| <!-- Spring容器 --> <!-- 配置Spring配置文件路径 --> <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>  classpath\*:\*\*/\*.service.xml  </param-value> </context-param> <!-- 配置Spring上下文监听器 --> <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class> </listener> |

## 字符编码过滤器

|  |
| --- |
| <!-- 配置Spring字符编码过滤器 --> <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param> </filter> <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern> </filter-mapping> |

## Spring mvc核心控制器

|  |
| --- |
| <!-- Spring MVC 核心控制器 DispatcherServlet 配置 --> <servlet>  <servlet-name>dispatcher</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- 设置spring mvc配置文件位置 -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath\*:springmvc/\*.controller.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup> </servlet> <servlet-mapping>  <servlet-name>dispatcher</servlet-name>  <!-- 拦截所有的请求,交给DispatcherServlet处理 -->  <url-pattern>/</url-pattern> </servlet-mapping> |

## Druid监控

|  |
| --- |
| <!-- druid web 监控 --> <servlet>  <servlet-name>DruidStatView</servlet-name>  <servlet-class>com.alibaba.druid.support.http.StatViewServlet</servlet-class> </servlet> <servlet-mapping>  <servlet-name>DruidStatView</servlet-name>  <url-pattern>/druid/\*</url-pattern> </servlet-mapping> |

## Logback日志

|  |
| --- |
| *<!--logback监听配置-->* <**context-param**>  <**param-name**>logbackConfigLocation</**param-name**>  <**param-value**>classpath:logback/logback.xml</**param-value**> </**context-param**> <**listener**>  <**listener-class**>ch.qos.logback.ext.spring.web.LogbackConfigListener</**listener-class**> </**listener**> |

# Spring配置

## basic.service.xml

|  |
| --- |
| <!-- 自动扫描路径 ,将带有注解的类纳入spring容器管理，排除Controller的类，交由spring mvc的上下文初始化 --> <context:component-scan base-package="com.fsindustry.cime.cimebm">  <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/> </context:component-scan>  <!-- 配置使Spring采用CGLIB代理 --> <aop:aspectj-autoproxy proxy-target-class="true" /> |

## Datasource.service.xml

|  |
| --- |
| <!-- 引入配置jdbc配置文件 --> <context:property-placeholder file-encoding="UTF-8"  location="classpath:jdbc.properties"/>  <!-- dataSource配置 --> <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init"  destroy-method="close" >  <!-- 基本属性 url、user、password -->  <property name="url" value="${jdbc.url}"/>  <property name="username" value="${jdbc.username}"/>  <property name="password" value="${jdbc.password}"/>   <!-- 配置初始化大小、最小、最大 -->  <property name="initialSize" value="${ds.initialSize}"/>  <property name="minIdle" value="${ds.minIdle}"/>  <property name="maxActive" value="${ds.maxActive}"/>   <!-- 配置获取连接等待超时的时间 -->  <property name="maxWait" value="${ds.maxWait}"/>   <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->  <property name="timeBetweenEvictionRunsMillis" value="${ds.timeBetweenEvictionRunsMillis}"/>   <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->  <property name="minEvictableIdleTimeMillis" value="${ds.minEvictableIdleTimeMillis}"/>   <property name="validationQuery" value="${ds.validationQuery}"/>  <property name="testWhileIdle" value="${ds.testWhileIdle}"/>  <property name="testOnBorrow" value="${ds.testOnBorrow}"/>  <property name="testOnReturn" value="${ds.testOnReturn}"/>   <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->  <property name="poolPreparedStatements" value="${ds.poolPreparedStatements}"/>  <property name="maxPoolPreparedStatementPerConnectionSize" value="${ds.maxPoolPreparedStatementPerConnectionSize}"/>   <!-- 配置监控统计拦截的filters -->  <property name="filters" value="${ds.filters}"/> </bean>  <!-- 配置session factory --> <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <!-- 配置数据源 -->  <property name="dataSource" ref="dataSource"/>  <!-- mybatis配置文件位置 -->  <property name="configLocation" value="classpath:mybatis/mybatis-config.xml"/>  <!-- mybatis映射文件位置 -->  <property name="mapperLocations" value="classpath\*:com/fsindustry/\*\*/\*.mapper.xml"/> </bean>  <!-- 配置mybatis接口映射 --> <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  <property name="basePackage" value="com.fsindustry.\*\*.dao"/>  <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/> </bean>  <!-- 配置事务管理器 --> <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <property name="dataSource" ref="dataSource"/> </bean>  <!-- 配置事务异常封装 --> <bean id="persistenceExceptionTranslationPostProcessor"  class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"/>  <!-- 声明式容器事务管理 ,transaction-manager指定事务管理器为transactionManager --> <tx:advice id="txAdvice" transaction-manager="transactionManager">  <tx:attributes>  <!-- 对insert,update,delete 开头的方法进行事务管理,只要有异常就回滚 -->  <tx:method name="insert\*" propagation="REQUIRED" rollback-for="java.lang.Throwable"/>  <tx:method name="update\*" propagation="REQUIRED" rollback-for="java.lang.Throwable"/>  <tx:method name="delete\*" propagation="REQUIRED" rollback-for="java.lang.Throwable"/>  <!-- select,count开头的方法,开启只读,提高数据库访问性能 -->  <tx:method name="select\*" propagation="NOT\_SUPPORTED" read-only="true"/>  <tx:method name="count\*" propagation="NOT\_SUPPORTED" read-only="true"/>  <!-- 对其他方法 使用默认的事务管理 -->  <tx:method name="\*" read-only="true"/>  </tx:attributes> </tx:advice>  <!-- 事务 aop 配置 --> <aop:config expose-proxy="true">  <aop:pointcut id="txPointcut" expression="execution(\* com.fsindustry.\*.service.\*.\*(..))"/>  <aop:advisor pointcut-ref="txPointcut" advice-ref="txAdvice"/> </aop:config>  <!-- 启用对事务注解的支持 --> <tx:annotation-driven transaction-manager="transactionManager"/> |

## Thread.service.xml

|  |
| --- |
| <!-- spring线程池配置 --> <bean id="threadPoolTaskExecutor" class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">  <!-- 核心线程数，默认为1 -->  <property name="corePoolSize" value="10" />  <!-- 最大线程数，默认为Integer.MAX\_VALUE -->  <property name="maxPoolSize" value="50" />  <!-- 队列最大长度，一般需要设置值>=notifyScheduledMainExecutor.maxNum；默认为Integer.MAX\_VALUE -->  <property name="queueCapacity" value="1000" />  <!-- 线程池维护线程所允许的空闲时间，默认为60s -->  <property name="keepAliveSeconds" value="120" />  <!-- 线程池对拒绝任务（无线程可用）的处理策略，目前只支持AbortPolicy、CallerRunsPolicy；默认为后者 -->  <property name="rejectedExecutionHandler">  <!-- AbortPolicy:直接抛出java.util.concurrent.RejectedExecutionException异常 -->  <!-- CallerRunsPolicy:主线程直接执行该任务，执行完之后尝试添加下一个任务到线程池中，可以有效降低向线程池内添加任务的速度 -->  <!-- DiscardOldestPolicy:抛弃旧的任务、暂不支持；会导致被丢弃的任务无法再次被执行 -->  <!-- DiscardPolicy:抛弃当前任务、暂不支持；会导致被丢弃的任务无法再次被执行 -->  <bean class="java.util.concurrent.ThreadPoolExecutor$CallerRunsPolicy" />  </property> </bean> |

# Spring mvc配置

|  |
| --- |
| <!-- 扫描controller（controller层注入） --> <context:component-scan base-package="com.fsindustry.cime.cimebm.sample.web"/>  <!-- 会自动注册DefaultAnnotationHandlerMapping与AnnotationMethodHandlerAdapter 两个bean,是spring MVC为@Controllers分发请求所必须的 --> <!-- 指定自己定义的validator --> <!-- <mvc:annotation-driven validator="validator" /> -->  <!-- 1.HandlerMapping 表示将请求的URL和Bean名字映射，如URL为 “上下文/hello”，则Spring配置文件必须有一个名字为“/hello”的Bean，上下文默认忽略 --> <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />  <!-- 2.HandlerAdapter 表示所有实现了org.springframework.web.servlet.mvc.Controller接口的Bean可以作为Spring Web   MVC中的业务控制器 --> <bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />  <!-- 3.视图解析器:定义跳转的文件的前后缀，将逻辑视图名称映射为物理视图文件名称 --> <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix" value="/WEB-INF/view/" />  <property name="suffix" value=".jsp" /> <!--可为空,方便实现自已的依据扩展名来选择视图解释类的逻辑 --> </bean>  <!-- 业务控制器样例 --> <bean name="/hello" class="com.fsindustry.cime.cimebm.sample.HelloWorldController" /> |

# mybatis-config.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">  <!-- 在配置文件中的位置必须符合要求，否则会报错，顺序如下: properties?, settings?, typeAliases?, typeHandlers?, objectFactory?,objectWrapperFactory?,   plugins?, environments?, databaseIdProvider?, mappers? --> <configuration>   <!-- 参数设置 -->  <settings>  <!-- 配置全局性 cache 的 ( 开 / 关) default:true -->  <setting name="cacheEnabled" value="true" />   <!-- 设置本地缓存范围 session:就会有数据的共享 statement:语句范围 (这样就不会有数据的共享 ) defalut:session -->  <setting name="localCacheScope" value="SESSION" />   <!-- 是否使用 懒加载 关联对象 同 hibernate中的延迟加载 一样 default:true -->  <setting name="lazyLoadingEnabled" value="true" />   <!-- [当对象使用延迟加载时 属性的加载取决于能被引用到的那些延迟属性,否则,按需加载(需要的是时候才去加载)] -->  <setting name="aggressiveLazyLoading" value="true" />   <!-- 是否允许单条sql 返回多个数据集 (取决于驱动的兼容性) default:true -->  <setting name="multipleResultSetsEnabled" value="true" />   <!-- 这是默认的执行类型 SIMPLE :简单 REUSE:执行器可能重复使用prepared statements 语句 BATCH:执行器可以重复执行语句和批量更新 -->  <setting name="defaultExecutorType" value="BATCH" />   <!-- [是否 启用 数据中 A\_column 自动映射 到 java类中驼峰命名的属性 default:fasle] -->  <setting name="mapUnderscoreToCamelCase" value="false" />   <!-- 设置但JDBC类型为空时,某些驱动程序 要指定值,default:other -->  <setting name="jdbcTypeForNull" value="NULL" />   </settings>   <!-- 别名映射 -->  <typeAliases>  <package name="com.fsindustry.cime.cimebm" />  </typeAliases>   <!-- 插件配置 -->  <plugins>  <!-- com.github.pagehelper为PageHelper类所在包名 -->  <plugin interceptor="com.github.pagehelper.PageHelper">  <!-- 数据库方言 -->  <property name="dialect" value="mysql" />   <!-- 设置为true时，会将RowBounds第一个参数offset当成pageNum页码使用 -->  <!-- 和startPage中的pageNum效果一样 -->  <!-- 该参数默认为false -->  <property name="offsetAsPageNum" value="true" />   <!-- 设置为true时，使用RowBounds分页会进行count查询 -->  <!-- 该参数默认为false -->  <property name="rowBoundsWithCount" value="true" />   <!-- 设置为true时，如果pageSize=0或者RowBounds.limit = 0就会查询出全部的结果 -->  <!-- （相当于没有执行分页查询，但是返回结果仍然是Page类型） -->  <property name="pageSizeZero" value="true" />   <!-- 3.3.0版本可用 - 分页参数合理化，默认false禁用 -->  <!-- 启用合理化时，如果pageNum<1会查询第一页，如果pageNum>pages会查询最后一页 -->  <!-- 禁用合理化时，如果pageNum<1或pageNum>pages会返回空数据 -->  <property name="reasonable" value="true" />   <!-- 3.5.0版本可用 - 为了支持startPage(Object params)方法 -->  <!-- 增加了一个`params`参数来配置参数映射，用于从Map或ServletRequest中取值 -->  <!-- 可以配置pageNum,pageSize,count,pageSizeZero,reasonable,不配置映射的用默认值 -->  <property name="params"  value="pageNum=start;pageSize=limit;pageSizeZero=zero;reasonable=heli;count=contsql" />  </plugin>  </plugins>  </configuration> |