## 新东西

* 1. font-awesome-4.7.0-dist
  2. waves-0.7.5-dist
  3. jquery-confirm-3.3.0.dist

## 创建maven项目

* 1. 使用空模版
  2. 完整目录结构
  3. 创建web模块（Module）
  4. 创建artifict
  5. 配置Tomcat

## 添加依赖

* 1. junit

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 <scope>test</scope>  
</dependency>

* 1. commos

<dependency>  
 <groupId>commons-io</groupId>  
 <artifactId>commons-io</artifactId>  
 <version>2.6</version>  
</dependency>  
<dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 <version>3.7</version>  
</dependency>  
<dependency>  
 <groupId>commons-fileupload</groupId>  
 <artifactId>commons-fileupload</artifactId>  
 <version>1.3.3</version>  
 <exclusions>  
 <exclusion>  
 <groupId>commons-io</groupId>  
 <artifactId>commons-io</artifactId>  
 </exclusion>  
 </exclusions>  
</dependency>  
<dependency>  
 <groupId>commons-logging</groupId>  
 <artifactId>commons-logging</artifactId>  
 <version>1.2</version>  
</dependency>

* 1. spring

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-expression</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-orm</artifactId>  
 <version>${spring.framework.version}</version>  
</dependency>

* 1. aspectj

<dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>${aspectj.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 <version>${aspectj.version}</version>  
</dependency>

* 1. mybatis

<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.4.5</version>  
</dependency>  
<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>1.3.1</version>  
</dependency>

<dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 <version>5.1.2</version>  
</dependency>

* 1. c3p0

<dependency>  
 <groupId>com.mchange</groupId>  
 <artifactId>c3p0</artifactId>  
 <version>0.9.5.2</version>  
</dependency>  
<dependency>  
 <groupId>com.mchange</groupId>  
 <artifactId>mchange-commons-java</artifactId>  
 <version>0.2.12</version>  
</dependency>

* 1. mysql-connector-java

<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.45</version>  
</dependency>

* 1. servlet-api

<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.0</version>  
</dependency>

* 1. jackson

<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>${jackson.version}</version>  
</dependency>

* 1. jstl

<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
</dependency>

* 1. logback

<dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>${logback.version}</version>  
</dependency>  
<dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-core</artifactId>  
 <version>${logback.version}</version>  
</dependency>  
<dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.25</version>  
</dependency>

## 添加插件

* 1. jetty-maven-plugin

<plugin>  
 <groupId>org.eclipse.jetty</groupId>  
 <artifactId>jetty-maven-plugin</artifactId>  
 <version>9.4.8.v20171121</version>  
 <configuration>  
 <scanIntervalSeconds>0</scanIntervalSeconds>  
  
 <httpConnector>  
 <port>8888</port>  
 </httpConnector>  
 </configuration>  
</plugin>

* 1. mybatis-generator-maven-plugin

<plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.6</version>  
</plugin>

## 添加配置文件

* 1. spring-mvc.xml

<context:component-scan base-package="com.swp"/>  
<!-- 导入spring-beans 配置 -->  
<import resource="classpath:spring/config/spring-beans.xml"/>  
  
<!-- 开启面向切面代理（自定义切面） -->  
<aop:aspectj-autoproxy/>  
  
<!-- 自动注册 DefaultAnnotationHandlerMapping 与 AnnotationMethodHandlerAdapter 两个bean，解决了@Controller注解的使用前提配置 -->  
<mvc:annotation-driven/>  
  
<!-- 根目录"／"对应页面 index.jsp 后缀名 .jsp 由视图解析器添加 -->  
<mvc:view-controller path="/" view-name="/index"/>  
  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="prefix" value="/WEB-INF/views/"/>  
 <property name="suffix" value=".jsp"/>  
 <property name="contentType" value="text/html; charset=utf-8"/>  
</bean>  
  
<!-- 返回json需导入jackson-annotations.jar ,jackson-core.jar, jackson-databind.jar -->  
<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>  
<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">  
 <property name="messageConverters">  
  
 <list>  
 <!-- 配置返回字符串编码 -->  
 <bean class="org.springframework.http.converter.StringHttpMessageConverter">  
 <property name="supportedMediaTypes">  
 <list>  
 <value>text/html;charset=UTF-8</value>  
 <value>application/json;charset=UTF-8</value>  
 </list>  
 </property>  
 </bean>  
  
 <!-- 配置json转换器 -->  
 <bean class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">  
 <property name="supportedMediaTypes">  
 <list>  
 <value>text/html;charset=UTF-8</value>  
 <value>application/json;charset=UTF-8</value>  
 </list>  
 </property>  
 </bean>  
  
 </list>  
  
 </property>  
  
</bean>  
  
<!-- 配置spring静态资源文件 两种方案 -->  
<!-- 一、将静态资源的处理经由Spring MVC框架交会Web应用服务器默认的defaultServlet处理-->  
<mvc:default-servlet-handler/>  
<!-- 二、把资源映射到ResourceHttpRequestHandler进行处理 -->  
<!--<mvc:resources mapping="/resources/\*\*" location="/resources"/>-->  
  
<!-- spingMVC 异常处理（由于404异常无法处理，404由web.xml处理） -->  
<bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">  
 <property name="exceptionMappings">  
 <props>  
 <prop key="java.lang.Exception">redirect:/WEB-INF/views/error.jsp</prop>  
 </props>  
 </property>  
 <property name="defaultErrorView" value="redirect:/WEB-INF/views/error.jsp"/>  
 <property name="defaultStatusCode" value="500"/>  
</bean>

* 1. applicationContext.xml

<context:component-scan base-package="com.swp"/>  
  
<!-- 加载properties -->  
<context:property-placeholder location="classpath:properties/db.properties"/>  
<!-- 加载配置文件 -->  
<import resource="classpath:spring/config/spring-\*.xml"/>

* 1. spring-beans.xml
  2. spring-mybatis.xml

<!-- 1.配置数据源 -->  
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">  
 <property name="user" value="${default.db.username}"/>  
 <property name="password" value="${default.db.password}"/>  
 <property name="driverClass" value="${default.db.driverClassName}"/>  
 <property name="jdbcUrl" value="${default.db.url}"/>  
  
 <property name="minPoolSize" value="5" />  
 <property name="maxPoolSize" value="20" />  
 <property name="initialPoolSize" value="10" />  
</bean>  
  
<!-- 2.SqlSessionFactory -->  
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  
 <property name="dataSource" ref="dataSource"/>  
 <property name="mapperLocations">  
 <list>  
 <value>classpath:com/swp/\*\*/\*Mapper.xml</value>  
 <value>classpath:mybatis/\*\*/\*Mapper.xml</value>  
 </list>  
 </property>  
</bean>  
  
<!-- 3 sqlSessionTemplate-->  
<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">  
 <constructor-arg ref="sqlSessionFactory"/>  
</bean>  
  
<!-- 扫描Mapper -->  
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <property name="basePackage" value="com.swp.module.product.mapper" />  
 <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />  
</bean>  
  
<!-- 4 事务管理器-->  
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource"/>  
</bean>  
  
<!-- 5 配置事务的传播行为 -->  
<tx:advice id="txAdvice" transaction-manager="transactionManager">  
 <tx:attributes>  
 <tx:method name="\*get" read-only="true"/>  
 <tx:method name="\*" propagation="REQUIRED"/>  
 </tx:attributes>  
</tx:advice>  
  
<aop:config>  
 <aop:pointcut id="pointCut" expression="execution(\* com.swp..\*.controller.\*.\*(..))"/>  
 <aop:advisor advice-ref="txAdvice" pointcut-ref="pointCut"/>  
</aop:config>  
  
<tx:annotation-driven transaction-manager="transactionManager"/>

* 1. db.properties

default.db.username=root  
default.db.password=101088  
default.db.driverClassName=com.mysql.jdbc.Driver  
default.db.url=jdbc:mysql:///demo?useUnicode=true&characterEncoding=utf8&useSSL=false  
  
  
db.initialSize=1  
db.minIdle=1  
db.maxActive=20  
db.maxWait=60000  
db.timeBetweenEvictionRunsMillis=60000  
db.minEvictableIdleTimeMillis=30000  
  
db.validationQuery=SELECT 'x'  
db.testWhileIdle=true  
db.testOnBorrow=false  
db.testOnReturn=false  
  
db.poolPreparedStatements=true  
db.maxPoolPreparedStatementPerConnectionSize=20  
db.filters=stat,wall,slf4j

* 1. web.xml

<!-- 过滤器解决中文乱码问题、强制编码UTF-8 -->  
<filter>  
 <filter-name>characterEncodingFilter</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>forceRequestEncoding</param-name>  
 <param-value>true</param-value>  
 </init-param>  
</filter>  
<filter-mapping>  
 <filter-name>characterEncodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
</filter-mapping>  
  
<servlet>  
 <servlet-name>spring-mvc</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring/spring-mvc.xml</param-value>  
 </init-param>  
</servlet>  
<servlet-mapping>  
 <servlet-name>spring-mvc</servlet-name>  
 <url-pattern>/</url-pattern>  
</servlet-mapping>  
  
<listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>  
<context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring/applicationContext.xml</param-value>  
</context-param>  
  
<!-- 配置session 失效时间 -->  
<session-config>  
 <session-timeout>30</session-timeout>  
</session-config>  
  
<!-- 配置欢迎页 -->  
<welcome-file-list>  
 <welcome-file>/WEB-INF/views/index.jsp</welcome-file>  
</welcome-file-list>  
  
<!-- 配置错误页 -->  
<error-page>  
 <error-code>404</error-code>  
 <location>/WEB-INF/views/error.jsp</location>  
</error-page>

(mybatis-generator的使用)

* 1. mbg.properties

mbg.jar=/Users/shiwanpeng/.m2/repository/mysql/mysql-connector-java/5.1.45/mysql-connector-java-5.1.45.jar  
mbg.path=/Users/shiwanpeng/Documents/GitHub/swp-blog/src/main  
  
mbg.username=root  
mbg.password=101088  
mbg.driverClassName=com.mysql.jdbc.Driver  
mbg.url=jdbc:mysql:///test?useUnicode=true&characterEncoding=utf8&useSSL=false

* 1. generatorConfig.xml

<generatorConfiguration>  
  
 <!-- 引入 properties 配置文件 -->  
 <properties resource="properties/mbg.properties"/>  
  
 <!-- 数据库链接驱动包 -->  
 <classPathEntry location="${mbg.jar}" />  
  
 <context id="MySqlTables" targetRuntime="MyBatis3">  
  
 <!-- 生成的Java文件的编码 -->  
 <property name="javaFileEncoding" value="UTF-8" />  
  
 <!-- 格式化java代码 -->  
 <property name="javaFormatter" value="org.mybatis.generator.api.dom.DefaultJavaFormatter" />  
  
 <!-- 格式化XML代码 -->  
 <property name="xmlFormatter" value="org.mybatis.generator.api.dom.DefaultXmlFormatter" />  
  
 <!-- 为生成的Java模型类添加序列化接口 -->  
 <plugin type="org.mybatis.generator.plugins.SerializablePlugin" />  
  
 <!-- 为生成的Java模型创建一个toString方法 -->  
 <plugin type="org.mybatis.generator.plugins.ToStringPlugin" />  
  
 <jdbcConnection driverClass="${mbg.driverClassName}" connectionURL="${mbg.url}" userId="${mbg.username}" password="${mbg.password}" />  
  
 <!-- java类型处理器 用于处理DB中的类型到Java中的类型，默认使用JavaTypeResolverDefaultImpl； 注意一点，默认会先尝试使用Integer，Long，Short等来对应DECIMAL和 NUMERIC数据类型； -->  
 <javaTypeResolver>  
 <property name="forceBigDecimals" value="false" />  
 </javaTypeResolver>  
  
 <!-- 实体类 -->  
 <javaModelGenerator targetPackage="com.swp.module.product.model" targetProject="${mbg.path}/java">  
 <property name="enableSubPackages" value="false" />  
 <property name="trimStrings" value="true" />  
 </javaModelGenerator>  
  
 <!-- \*Mapper.xml -->  
 <sqlMapGenerator targetPackage="mybatis.module.product" targetProject="${mbg.path}/resources">  
 <property name="enableSubPackages" value="false" />  
 </sqlMapGenerator>  
  
 <!-- \*Mapper.java-->  
 <javaClientGenerator targetPackage="com.swp.module.product.mapper" targetProject="${mbg.path}/java" type="XMLMAPPER">  
 <property name="enableSubPackages" value="false" />  
 </javaClientGenerator>  
  
 <table schema="" tableName="TB\_PRODUCT" />  
  
 </context>  
</generatorConfiguration>

* 1. logback.xml

<configuration scan="true" scanPeriod="60 seconds" debug="false">  
  
 <!-- 引入 properties 配置文件（定义日志文件的存储地址 勿在 LogBack 的配置中使用相对路径） -->  
 <property resource="properties/log.properties"/>  
 <property resource="properties/db.properties"/>  
  
 <!-- 控制台输出 -->  
 <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
 <encoder>  
 <!--格式化输出：%d表示日期，%thread表示线程名，%-5level：级别从左显示5个字符宽度%msg：日志消息，%n是换行符 -->  
 <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>  
 </encoder>  
 </appender>  
  
 <!-- ERROR 级别日志 -->  
 <!-- 滚动记录文件，先将日志记录到指定文件，当符合某个条件时，将日志记录到其他文件 RollingFileAppender -->  
 <appender name="ERROR" class="ch.qos.logback.core.rolling.RollingFileAppender">  
 <!-- 过滤器，只记录WARN级别的日志 -->  
 <filter class="ch.qos.logback.classic.filter.LevelFilter">  
 <level>ERROR</level>  
 <onMatch>ACCEPT</onMatch>  
 <onMismatch>DENY</onMismatch>  
 </filter>  
 <!-- 最常用的滚动策略，它根据时间来制定滚动策略.既负责滚动也负责出发滚动 -->  
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
 <!--日志输出位置 可相对、和绝对路径 -->  
 <fileNamePattern>${log.dir}/%d{yyyy-MM-dd}/${log.name}.error.log</fileNamePattern>  
 <!-- 可选节点，控制保留的归档文件的最大数量，超出数量就删除旧文件假设设置每个月滚动，且<log.maxHistory>是6，  
 则只保存最近6个月的文件，删除之前的旧文件。注意，删除旧文件是，那些为了归档而创建的目录也会被删除-->  
 <log.maxHistory>${log.maxHistory}</log.maxHistory>  
 </rollingPolicy>  
  
 <!-- 按照固定窗口模式生成日志文件，当文件大于20MB时，生成新的日志文件。窗口大小是1到3，当保存了3个归档文件后，将覆盖最早的日志。  
 <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">  
 <fileNamePattern>${log.dir}/%d{yyyy-MM-dd}/.log.zip</fileNamePattern>  
 <minIndex>1</minIndex>  
 <maxIndex>3</maxIndex>  
 </rollingPolicy> -->  
 <!-- 查看当前活动文件的大小，如果超过指定大小会告知RollingFileAppender 触发当前活动文件滚动  
 <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">  
 <maxFileSize>5MB</maxFileSize>  
 </triggeringPolicy> -->  
  
 <encoder>  
 <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%n</pattern>  
 </encoder>  
 <!-- 日志文件最大的大小 -->  
 <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">  
 <MaxFileSize>${log.maxSize}</MaxFileSize>  
 </triggeringPolicy>  
 </appender>  
  
 <!-- WARN 级别日志 -->  
 <appender name="WARN" class="ch.qos.logback.core.rolling.RollingFileAppender">  
 <!-- 过滤器，只记录WARN级别的日志 -->  
 <filter class="ch.qos.logback.classic.filter.LevelFilter">  
 <level>WARN</level>  
 <onMatch>ACCEPT</onMatch>  
 <onMismatch>DENY</onMismatch>  
 </filter>  
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
 <!-- 日志文件输出的文件名 使用 ${user.dir} 或 ${catalina.base} -->  
 <fileNamePattern>${log.dir}/%d{yyyy-MM-dd}/${log.name}.warn.log</fileNamePattern>  
 <!-- 日志最大的历史天数 -->  
 <log.maxHistory>${log.maxHistory}</log.maxHistory>  
 </rollingPolicy>  
 <encoder>  
 <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%n</pattern>  
 </encoder>  
 <!-- 日志文件最大的大小 -->  
 <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">  
 <MaxFileSize>${log.maxSize}</MaxFileSize>  
 </triggeringPolicy>  
 </appender>  
  
 <!-- INFO级别日志 -->  
 <appender name="INFO" class="ch.qos.logback.core.rolling.RollingFileAppender">  
 <!-- 过滤器，只记录INFO级别的日志 -->  
 <filter class="ch.qos.logback.classic.filter.LevelFilter">  
 <level>INFO</level>  
 <onMatch>ACCEPT</onMatch>  
 <onMismatch>DENY</onMismatch>  
 </filter>  
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
 <!-- 日志文件输出的文件名 使用 ${user.dir} 或 ${catalina.base} -->  
 <fileNamePattern>${log.dir}/%d{yyyy-MM-dd}/${log.name}.info.log</fileNamePattern>  
 <!-- 日志最大的历史天数 -->  
 <log.maxHistory>${log.maxHistory}</log.maxHistory>  
 </rollingPolicy>  
 <encoder>  
 <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%n</pattern>  
 </encoder>  
 <!-- 日志文件最大的大小 -->  
 <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">  
 <MaxFileSize>${log.maxSize}</MaxFileSize>  
 </triggeringPolicy>  
 </appender>  
  
 <!-- DEBUG级别日志 appender -->  
 <appender name="DEBUG" class="ch.qos.logback.core.rolling.RollingFileAppender">  
 <!-- 过滤器，只记录DEBUG级别的日志 -->  
 <filter class="ch.qos.logback.classic.filter.LevelFilter">  
 <level>DEBUG</level>  
 <onMatch>ACCEPT</onMatch>  
 <onMismatch>DENY</onMismatch>  
 </filter>  
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
 <!-- 日志文件输出的文件名 使用 ${user.dir} 或 ${catalina.base} -->  
 <fileNamePattern>${log.dir}/%d{yyyy-MM-dd}/${log.name}.debug.log  
 </fileNamePattern>  
 <!-- 日志最大的历史天数 -->  
 <log.maxHistory>${log.maxHistory}</log.maxHistory>  
 </rollingPolicy>  
 <encoder>  
 <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%n</pattern>  
 </encoder>  
 <!-- 日志文件最大的大小 -->  
 <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">  
 <MaxFileSize>${log.maxSize}</MaxFileSize>  
 </triggeringPolicy>  
 </appender>  
  
 <!-- TRACE级别日志 -->  
 <appender name="TRACE" class="ch.qos.logback.core.rolling.RollingFileAppender">  
 <!-- 过滤器，只记录ERROR级别的日志 -->  
 <filter class="ch.qos.logback.classic.filter.LevelFilter">  
 <level>TRACE</level>  
 <onMatch>ACCEPT</onMatch>  
 <onMismatch>DENY</onMismatch>  
 </filter>  
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
 <!-- 按天回滚 daily -->  
 <fileNamePattern>${log.dir}/%d{yyyy-MM-dd}/${log.name}.trace.log</fileNamePattern>  
 <!-- 日志最大的历史天数 -->  
 <log.maxHistory>${log.maxHistory}</log.maxHistory>  
 </rollingPolicy>  
 <encoder>  
 <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%n</pattern>  
 </encoder>  
 <!-- 日志文件最大的大小 -->  
 <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">  
 <MaxFileSize>${log.maxSize}</MaxFileSize>  
 </triggeringPolicy>  
 </appender>  
  
 <!--日志异步到数据库 -->  
 <!--<appender name="DB" class="ch.qos.logback.classic.db.DBAppender">-->  
 <!--<filter class="ch.qos.logback.classic.filter.LevelFilter">-->  
 <!--<level>WARN</level>-->  
 <!--<onMatch>ACCEPT</onMatch>-->  
 <!--<onMismatch>DENY</onMismatch>-->  
 <!--</filter>-->  
 <!--&lt;!&ndash;日志异步到数据库 &ndash;&gt;-->  
 <!--<connectionSource class="ch.qos.logback.core.db.DataSourceConnectionSource">-->  
 <!--&lt;!&ndash;连接池 &ndash;&gt;-->  
 <!--<dataSource class="com.mchange.v2.c3p0.ComboPooledDataSource">-->  
 <!--<url>${default.db.url}</url>-->  
 <!--<username>${default.db.username}</username>-->  
 <!--<password>${default.db.password}</password>-->  
 <!--<driverClassName>${default.db.driverClassName}</driverClassName>-->  
 <!--</dataSource>-->  
 <!--</connectionSource>-->  
 <!--</appender>-->  
  
 <!-- 打印 sql 配置 -->  
 <logger name="java.sql.Connection" level="DEBUG"/>  
 <logger name="java.sql.Statement" level="DEBUG"/>  
 <logger name="java.sql.PreparedStatement" level="DEBUG"/>  
 <logger name="java.sql.ResultSet" level="DEBUG"/>  
 <!-- 为 mybatis sql 定制 -->  
 <logger name="com.ibatis" level="DEBUG"/>  
 <logger name="com.ibatis.common.jdbc.SimpleDataSource" level="DEBUG"/>  
 <logger name="com.ibatis.common.jdbc.ScriptRunner" level="DEBUG"/>  
 <logger name="org.mybatis.generator" level="DEBUG"/>  
 <logger name="com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate" level="DEBUG"/>  
  
 <!-- 为 Hibernate sql 定制 -->  
 <!--  
 <logger name="org.hibernate.type.descriptor.sql.BasicBinder" level="TRACE" />  
 <logger name="org.hibernate.type.descriptor.sql.BasicExtractor" level="DEBUG" />  
 <logger name="org.hibernate.SQL" level="DEBUG" />  
 <logger name="org.hibernate.engine.QueryParameters" level="DEBUG" />  
 <logger name="org.hibernate.engine.query.HQLQueryPlan" level="DEBUG" />  
 -->  
  
 <!-- 日志输出级别 -->  
 <root level="INFO">  
 <!-- 控制台输出 -->  
 <appender-ref ref="STDOUT"/>  
 <!-- 文件输出 -->  
 <appender-ref ref="ERROR"/>  
 <appender-ref ref="INFO"/>  
 <appender-ref ref="WARN"/>  
 <appender-ref ref="DEBUG"/>  
 <appender-ref ref="TRACE"/>  
 <!-- 数据库输出 -->  
 <!--<appender-ref ref="DB"/>-->  
 </root>  
  
</configuration>

## 登录

* 1. 自定义注解

@Documented  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*FIELD*)  
public @interface MapperInject {  
  
 */\*\*  
 \* 对象类型（默认 Object 则认为是 DelegateMapper 类型）  
 \*  
 \** ***@return*** *Class*<?> *对象类型</>  
 \*/* Class<?> value() default java.lang.Object.class;  
  
}

* 1. 创建DelegateMapper，实现最常规的CRUD操作, sqlsessionTemplate作为参数（注意@Autoworried注入的值不能在构造函数中使用）
  2. 自定义Mapper注入器，MapperInjector。实现了后置处理器接口BeanPostProcessor,在每一个Bean创建的使用都会执行。其中用到了反射技术。

@Override  
public Object postProcessBeforeInitialization(Object bean, String beanName) throws BeansException {  
 ReflectionUtils.*doWithFields*(bean.getClass(), new ReflectionUtils.FieldCallback() {  
 */\*\*  
 \* 通过注解（反射）创建Mapper实例  
 \** ***@param*** *field  
 \** ***@throws*** *IllegalArgumentException  
 \** ***@throws*** *IllegalAccessException  
 \*/* public void doWith(Field field) throws IllegalArgumentException, IllegalAccessException {  
 ReflectionUtils.*makeAccessible*(field);  
 if (field.isAnnotationPresent(MapperInject.class)){  
 MapperInject annotation = field.getAnnotation(MapperInject.class);  
 Class<?> clazz = annotation.value();  
 if ("DelegateMapper".equals(field.getType().getSimpleName())){  
 field.set(bean, new DelegateMapper(sqlSessionTemplate));  
 }else {  
 field.set(bean,sqlSessionTemplate.getMapper(clazz));  
 }  
 }  
 }  
 });  
 return bean;  
}

@Override  
public Object postProcessAfterInitialization(Object bean, String name) throws BeansException {  
 return bean;  
}

* 1. LoginUser 中日期属性的定义类型为：Date。Mapper.xml中定义的类型为：DATE（年月日），TIMESTAMP（年月日时分秒）。数据库中字段的类型：date（年月日），datetime（年月日时分秒）

## BaseSupport基础工具类

* 1. 判断是否为空（包括一般对象，字符串，List，Map，Set）
  2. 对象转换为String
  3. 获取UUID
  4. Base64编码，解码
  5. MD5加密
  6. 获取当前系统日期，时间戳转日期，日期转时间戳
  7. 从指定文件中读取数据
  8. 奖内容写入指定的文件中
  9. 读取配置properties 文件中的值

## ControllerSupport 控制器支持类

* 1. 获取mapper
  2. 获取delegateMapper
  3. 动态切换数据源
  4. 清除数据源
  5. 分页查询范围
  6. 分页结果集对象
  7. 消息返回对象
  8. 文件上传
  9. 文件下载

## BaseController 基本控制器

* 1. 获取会话session
  2. 获取请求request （HttpServletRequest）
  3. 获取服务器响应response （HttpServletResponse）
  4. 获取登录成功后session中存储的用户信息
  5. 异常处理（shiro）

## LoginFilter 对未登录的\*.jsp进行过滤（在web.xml中配置）

## LoginInterception 拦截未登录的请求（在spring-mvc中配置）

## Logger日志类的使用

* 1. 通过工厂类LoggonFactory获取
  2. 通过自定义注解@LogInject获取

## @Aspect 面向切面编程 LogAspect

* 1. @Pointcut 自定义切入点 （execution）
  2. @Before 前置通知
  3. @After 后置通知
  4. @AfterReturing 返回通知
  5. @AfterThrowing 异常通知
  6. @Around 环绕通知
  7. JoinPoint通用参数。
     1. Classname = JoinPoint.getTarget().getClass().getName()
     2. Method = JoinPoint.getSignature().getName()
     3. Parameters = Arrays.asList(JoinPoint.getArgs())